

International Advanced Level **Computer Science**

Sample Assessment Materials

Pearson Edexcel International Advanced Level in Computer Science (YCP01)

First teaching September 2026 | First examination June 2028

First certification from August 2028

Issue 1

Content

Introduction	1
Unit 3: Advanced principles of computer science	2
Question paper	2
Mark scheme	24
Assembly Language Instruction Set	46
Unit 4: Advanced practical programming and problem-solving	48
Question paper	48
Mark scheme	59
Programming Language Subset (PLS)	97



Introduction

The Pearson Edexcel International Advanced Level in Computer Science consists of the two IAS Units, 1 and 2, plus two IA2 Units, 3 and 4. It is designed for use in schools and colleges. It is part of a suite of International Advanced Level qualifications offered by Pearson.

These sample assessment materials have been developed to support this qualification and will be used as the benchmark to develop the assessment learners will take. The sample assessment material for International Advanced Subsidiary in Computer Science is available to download from the Pearson Edexcel International Advanced Level in Computer Science website: qualifications.pearson.com.

Onscreen sample assessment material for Unit 3 is available within our sandbox environment. It gives the opportunity to experience the platform and the functionalities available.

Go to the [Sandbox environment](#) > Click on “Register for an account” > Enter your email address as your UserID > Wait for an automatic email to confirm your email address > Proceed to log in.

The data files for Unit 4 are available to download from the Pearson Edexcel International Advanced Level in Computer Science website: qualifications.pearson.com.

Please check the examination details below before entering your candidate information

Candidate surname					Other names				
Centre Number					Candidate Number				

Pearson Edexcel International Advanced Level

Sample assessment material

Time 1 hour 30 minutes

Paper reference **WCP03/01**

Computer Science
International Advanced Level
Unit 3: Advanced Principles of Computer Science

You must have:
 Resource Booklet (enclosed)

Total Marks

Instructions

- Use **black** ink or ball-point pen.
- **Fill in the boxes** at the top of this page with your name, centre number and candidate number.
- Answer **all** questions.
- Answer the questions in the spaces provided
– *there may be more space than you need.*
- You are not allowed to use a calculator.

Information

- The total mark for this paper is 80.
- The marks for **each** question are shown in brackets
– *use this as a guide as to how much time to spend on each question.*

Advice

- Read each question carefully before you start to answer it.
- Try to answer every question.
- Check your answers if you have time at the end.

S85796A

©2026 Pearson Education Ltd.
1/1/1/1/1/1/1



Answer ALL questions. Write your answers in the spaces provided.

Some questions must be answered with a cross in a box ☒. If you change your mind about an answer, put a line through the box ☒ and then mark your new answer with a cross ☒.

SECTION A

Computer systems and data representation

1 Which type of processor architecture is used by a graphics processing unit (GPU)?

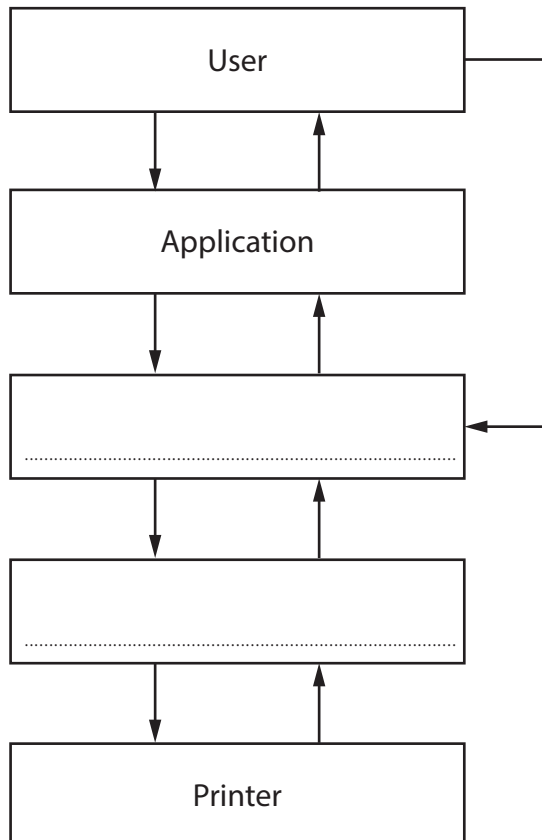
- ☐ A multiple instruction multiple data (MIMD)
- ☐ B multiple instruction single data (MISD)
- ☐ C single instruction multiple data (SIMD)
- ☐ D single instruction single data (SISD)

(Total for Question 1 = 1 mark)

2 The partially completed diagram shows how a user interacts with the components of a computer system.

Complete the diagram by adding the **two** missing components.

Diagram



(Total for Question 2 = 2 marks)

3 A software development company uses a virtual machine to write and test software.

(a) Define the term 'virtual machine'.

(2)

(b) Explain **one** way a virtual machine is suitable for this purpose.

(2)

(Total for Question 3 = 4 marks)

4 Explain **one** advantage of using Harvard architecture rather than von Neumann architecture.

(Total for Question 4 = 2 marks)

5 Most mobile phones use RISC architecture.

(a) State what RISC stands for.

(1)

(b) Explain **one** characteristic of RISC architecture that results in devices having longer battery life than devices using CISC architecture.

(2)

(Total for Question 5 = 3 marks)

6 A weather station records air temperature.

Describe how an analogue to digital converter (ADC) is used in this process.

(Total for Question 6 = 3 marks)

DO NOT WRITE IN THIS AREA

7 A processor uses a five-stage pipeline:

- instruction fetch (IF)
- instruction decode (ID)
- execute (EX)
- memory access (MEM)
- register write back (WB).

A program contains the five instructions A, B, C, D and E.

Complete the table to show how the program would execute using pipelining.

Table

	Clock cycle									
Pipeline stage	1	2	3	4	5	6	7	8	9	10
IF										
ID										
EX										
MEM										
WB										

(Total for Question 7 = 3 marks)

8 Explain **one** impact a branch instruction has on a pipeline.

.....

.....

.....

.....

.....

.....

(Total for Question 8 = 2 marks)

TOTAL FOR SECTION A = 20 MARKS

SECTION B

Emerging technologies and professional practice

9 Deep learning is a subset of machine learning.

Which statement best defines deep learning?

- ☐ **A** It analyses unlabelled data to find patterns and insights without human guidance.
- ☐ **B** It teaches systems to make actions in an environment to maximise rewards.
- ☐ **C** It uses labelled data sets to train algorithms to recognise patterns and predict outcomes.
- ☐ **D** It uses multilayered neural networks to simulate the complex decision-making power of the human brain.

(Total for Question 9 = 1 mark)

10 Complete the table by adding the quantum computing term for the **two** definitions.

Table

Quantum computing term	Definition
	The state in which a quantum particle or system can represent not just one possibility, but a combination of multiple possibilities.
	The basic unit of quantum information, which is the quantum version of the classic binary bit.

(Total for Question 10 = 2 marks)

11 Describe the process used to create a digital signature.

(Total for Question 11 = 3 marks)

12 A block chain contains two blocks: block 1 and block 2.

An attacker has modified the contents of block 1.

Explain **one** way unauthorised modification of block 1 can be detected in block 2.

(Total for Question 12 = 2 marks)

TOTAL FOR SECTION B = 8 MARKS

SECTION C

Programming languages

13 Define the term 'polymorphism' in object-oriented programming (OOP).

(Total for Question 13 = 2 marks)

14 The program is written using object-oriented programming (OOP).

Program

```
1 class Character():
2
3     def __init__(self, pName):
4         self._name = pName
5         self._strength = 10
6
7     def getStrength(self):
8         return self._strength
9
10    def setStrength(self, pStrength):
11        self._strength = pStrength
12
13    def display(self):
14        print("Name:", self._name)
15
16 class Warrior(Character):
17
18     def __init__(self, pName):
19         super().__init__(pName)
20         self._strength = 18
21         self.__status = "Warrior"
22
23     def display(self):
24         super().display()
25         print("Status:", self.__status)
```

DO NOT WRITE IN THIS AREA

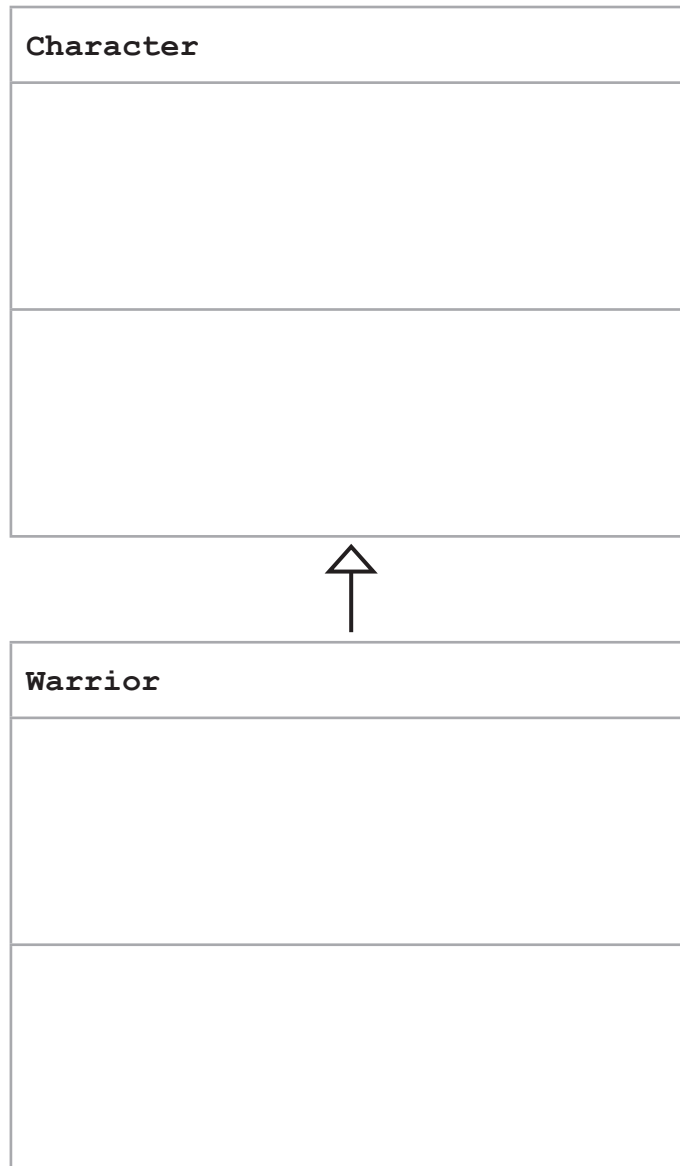
DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

Complete the class diagram for the program by adding:

- attributes
- methods
- access modifiers: private (-), protected (#), public (+).

Class diagram



(Total for Question 14 = 4 marks)

15 The program is written in assembly language.

Program

```
MOV R0, #6
MOV R1, #5
MOV R2, #0
MOV R3, #0
start:  CMP R1, #0
        BEQ end
        MOV R4, R1
        AND R4, R4, #1
        CMP R4, #1
        BNE skip
        LSL R5, R0, R2
        ADD R3, R3, R5
skip:   ADD R2, R2, #1
        LSR R1, R1, #1
        B start
end:    HALT
```

Complete the trace table using the program to:

- trace the execution of the program
- determine the function of the program.

You may not need to use all rows and cells.

DO NOT WRITE IN THIS AREA

R0	R1	R2	R3	R4	R5
Program function					

(Total for Question 15 = 6 marks)

TOTAL FOR SECTION C = 12 MARKS

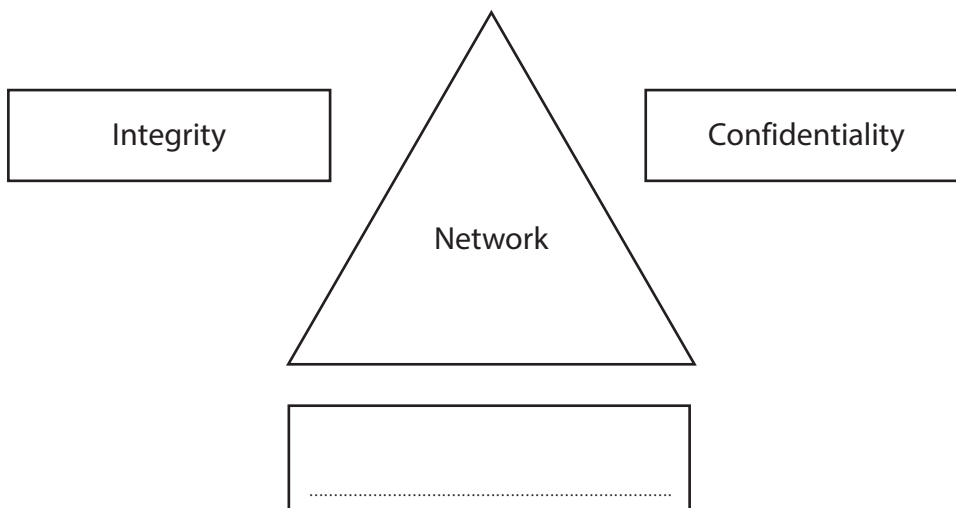
SECTION D

Network and cybersecurity

16 The partially completed diagram shows the CIA network security model.

Complete the diagram by adding the missing label.

Diagram



(Total for Question 16 = 1 mark)

17 State the name of the protocol used to translate www.pearson.com to an IP address.

(Total for Question 17 = 1 mark)

18 Four bytes are transmitted using the same type of parity.

Which byte contains a single bit that is corrupted during transmission?

- ☐ **A** 0000 1010
- ☐ **B** 0100 0100
- ☐ **C** 1001 1111
- ☐ **D** 1010 0100

(Total for Question 18 = 1 mark)

19 Describe how an actuator functions in an Internet of things (IoT) device.

(Total for Question 19 = 2 marks)

20 Explain **one** way the source IP address in an IP packet header is used.

(Total for Question 20 = 2 marks)

21 Explain **one** way using large prime numbers makes RSA encryption secure.

(Total for Question 21 = 2 marks)

TOTAL FOR SECTION D = 9 MARKS

SECTION E

Structuring data

22 Give **two** advantages of using an adjacency matrix rather than an adjacency list to represent a graph.

- 1
- 2

(Total for Question 22 = 2 marks)

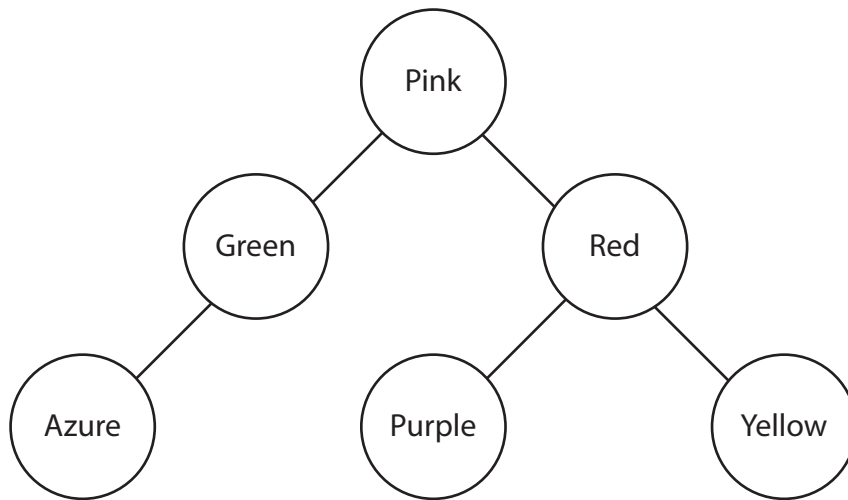
DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

23 The diagram shows a binary search tree.

Binary search tree



(a) Describe the process of searching for the value Purple in the binary search tree.

(4)

.....

.....

.....

.....

.....

.....

.....

.....

(b) Give the name of the traversal algorithm that would output the values in the nodes in the order Pink, Green, Azure, Red, Purple and Yellow.

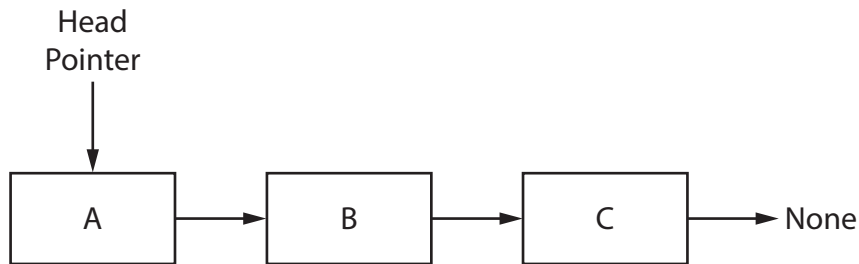
(1)

.....

(Total for Question 23 = 5 marks)

24 The linked list contains three values.

Linked list



The function `count()` takes a pointer as a parameter.

Outline how the function can use recursion to count the number of items shown in the linked list.

(Total for Question 24 = 4 marks)

TOTAL FOR SECTION E = 11 MARKS

SECTION F**Problem solving**

25 Which statement describes an algorithm with linear complexity?

- ☒ **A** It halves the number of items to process at each step.
- ☒ **B** It takes the same amount of time to execute regardless of the number of items to process.
- ☒ **C** Its run time doubles with each additional data item to process.
- ☒ **D** Its run time grows in direct proportion to the number of items to process.

(Total for Question 25 = 1 mark)

26 Complete the table to give the average case time complexity for the **two** operations.

Table

Operation	Average case
Binary search	
Merge sort	

(Total for Question 26 = 2 marks)

27 Outline how a pivot is used by a quick sort algorithm when performing partitioning.

(Total for Question 27 = 2 marks)

28 Explain **one** problem-solving technique that can be used to solve an intractable problem.

(Total for Question 28 = 2 marks)

DO NOT WRITE IN THIS AREA

29 The Karnaugh map represents a Boolean function.

Karnaugh map

		AB			
		00	01	11	10
CD	00	0	1	1	0
	01	1	0	1	0
	11	1	0	1	0
	10	0	1	1	0

Construct a Boolean expression for the function in the Karnaugh map as a simplified sum of products.

(Total for Question 29 = 4 marks)

30 Simplify this expression using Boolean algebra.

$$\overline{\overline{X} + \overline{Y}} + Y \cdot \overline{X}$$

Show your workings.

Answer

(Total for Question 30 = 3 marks)

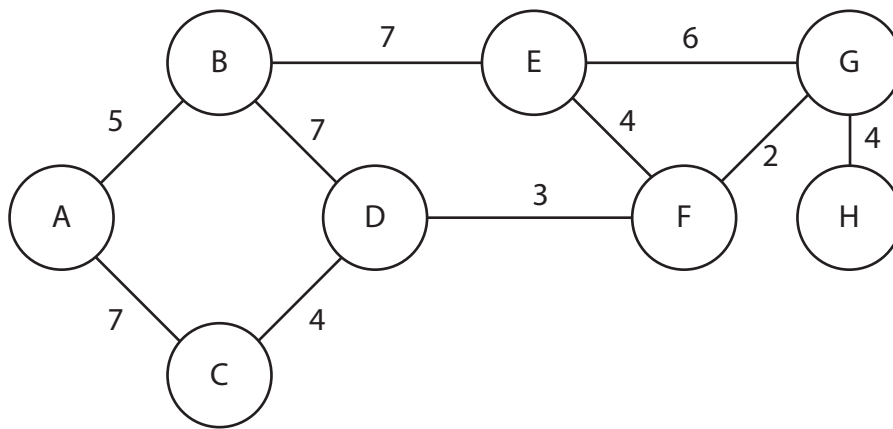
DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

- 31** The diagram shows the distance it takes to move between different locations in a computer game.

Locations in a computer game



Complete the table using Dijkstra's shortest path algorithm to:

- find the shortest path from location A to location H
- show the overall distance and the shortest path to reach location H.

Table

Node	Distance to node	Path to node
A		
B		
C		
D		
E		
F		
G		
H		
Distance		
Shortest path		

(Total for Question 31 = 6 marks)

TOTAL FOR SECTION F = 20 MARKS
TOTAL FOR PAPER = 80 MARKS

DO NOT WRITE IN THIS AREA

International Advanced Level in Computer Science

Unit 3: Advanced Principles of Computer Science

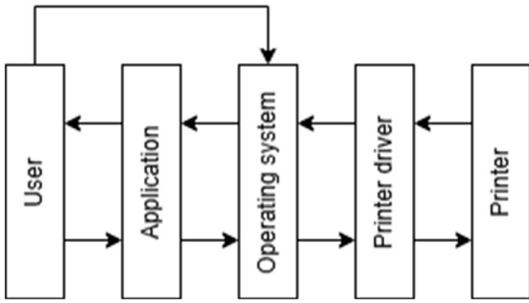
Mark Scheme

General marking guidance

- All candidates must receive the same treatment. Examiners must mark the first candidate in exactly the same way as they mark the last.
- Mark schemes should be applied positively. Candidates must be rewarded for what they have shown they can do rather than penalised for omissions.
- Examiners should mark according to the mark scheme not according to their perception of where the grade boundaries may lie.
- There is no ceiling on achievement. All marks on the mark scheme should be used appropriately.
- All the marks on the mark scheme are designed to be awarded. Examiners should always award full marks if deserved, i.e. if the answer matches the mark scheme. Examiners should also be prepared to award zero marks if the candidate's response is not worthy of credit according to the mark scheme.
- Where some judgement is required, mark schemes will provide the principles by which marks will be awarded and exemplification may be limited.
- When examiners are in doubt regarding the application of the mark scheme to a candidate's response, the team leader must be consulted.
- Crossed-out work should be marked UNLESS the candidate has replaced it with an alternative response.

This mark scheme includes colour coded text. Centres must ensure that all hard copies are printed in full colour so that assessment guidance is accurately conveyed.

Question number	Answer	Additional Guidance	Mark
1	The only correct answer is C A is not correct because MIMD is typically used for general parallel processing and distributed computing B is not correct because MISD has limited practical use in fault tolerant systems D is not correct because SISD is used for basic sequential computing tasks	The correct answer is: C Single Instruction Multiple Data (SIMD).	1

Question number	Answer	Additional Guidance	Mark
2	Award one mark for each of the following up to a maximum of two marks: • Operating system/OS (1) • Printer driver/Device driver (1) Exemplar answer:  <pre> graph TD User[User] <--> Application[Application] User <--> OS[Operating system] Application <--> OS OS <--> PD[Printer driver] PD <--> Printer[Printer] </pre>		2

Question number	Answer	Additional Guidance	Mark
3(a)	Award up to two marks for a linked definition that makes reference to: • An emulation of an existing computer system (1) running on a different physical computer system (1) • An emulation of a guest operating system (1) running on a different host operating system (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
3(b)	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks. • Testing will not harm the host machine (1) because the virtual machine acts as a sandbox / can easily be reset (1) • There is no need for multiple physical platforms (1) because multiple virtual machines for different platforms can be installed on the host (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
4	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks. • Performance is improved (1) because parallel access to data and instructions is possible (1) • Optimised/improved use of memory (1) because instruction and data memory sizes can be different (1) • Greater security from malicious code (1) because data cannot be executed as code / instructions cannot be over-written (1) Accept any other appropriate response.	Do not accept generic faster/quicker/more efficient without qualification.	2

Question number	Answer	Additional Guidance	Mark
5(a)	Award one mark for: • Reduced Instruction Set Computer (1)		1

Question number	Answer	Additional Guidance	Mark
5(b)	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks. • Devices using RISC architecture have a lower power consumption (than CISC) (1) because they use fewer transistors (1) • Devices using RISC architecture have a lower power consumption (than CISC) (1) because they require fewer complex decoding/logic components (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
6	Award one mark for each appropriate point given in the correct order, up to a maximum of three marks. • The analogue temperature signals are sampled at regular intervals (1) and these samples are then quantised to discrete values within a set range (1) and then encoded as a fixed number of bits (1) Accept any other appropriate response.		3

Question number	Answer	Additional Guidance	Mark																																																																													
7	Award one mark for each of the following up to a maximum of three marks: • Instruction A pipeline stages completed over clock cycle 1 to 5 (1) • Instruction B, C, D and E Instruction Fetch follows Instruction A Fetch in sequence in cycles 2 to 5 (1) • Remaining pipeline stages for instructions B, C, D and E follow A (1) Exemplar answer: <table><tr><th colspan="11">Clock cycle</th></tr><tr><th>Pipeline stage</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th></tr><tr><td>IF</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>ID</td><td></td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td></td><td></td><td></td><td></td></tr><tr><td>EX</td><td></td><td></td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td></td><td></td><td></td></tr><tr><td>MEM</td><td></td><td></td><td></td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td></td><td></td></tr><tr><td>WB</td><td></td><td></td><td></td><td></td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td></td></tr></table>	Clock cycle											Pipeline stage	1	2	3	4	5	6	7	8	9	10	IF	A	B	C	D	E						ID		A	B	C	D	E					EX			A	B	C	D	E				MEM				A	B	C	D	E			WB					A	B	C	D	E		Mark points colour coded: BP1 BP2 BP3	3
Clock cycle																																																																																
Pipeline stage	1	2	3	4	5	6	7	8	9	10																																																																						
IF	A	B	C	D	E																																																																											
ID		A	B	C	D	E																																																																										
EX			A	B	C	D	E																																																																									
MEM				A	B	C	D	E																																																																								
WB					A	B	C	D	E																																																																							

Question number	Answer	Additional Guidance	Mark
8	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks. - The pipeline would need to be reset/flushed (1) because the next instruction in the pipeline is not the next instruction that needs to be executed (1) - The contents of the pipeline would need to be discarded (1) because the instruction branched to is not the next instruction in the pipeline (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
9	The only correct answer is D A is not correct because it defines unsupervised learning B is not correct because it defines reinforcement learning C is not correct because it describes supervised learning	The correct answer is: D it uses multilayered neural networks to simulate the complex decision-making power of the human brain.	1

Question number	Answer	Additional Guidance	Mark						
10	Award one mark for each of the following up to a maximum of two marks: <table><tr><th>Quantum computing term</th><th>Definition</th></tr><tr><td>(Quantum) Superposition (1)</td><td>The state in which a quantum particle or system can represent not just one possibility, but a combination of multiple possibilities.</td></tr><tr><td>Qubit (1)</td><td>The basic unit of quantum information, which is the quantum version of the classic binary bit.</td></tr></table>	Quantum computing term	Definition	(Quantum) Superposition (1)	The state in which a quantum particle or system can represent not just one possibility, but a combination of multiple possibilities.	Qubit (1)	The basic unit of quantum information, which is the quantum version of the classic binary bit.		2
Quantum computing term	Definition								
(Quantum) Superposition (1)	The state in which a quantum particle or system can represent not just one possibility, but a combination of multiple possibilities.								
Qubit (1)	The basic unit of quantum information, which is the quantum version of the classic binary bit.								

Question number	Answer	Additional Guidance	Mark
11	Award one mark for each appropriate point given in the correct order, up to a maximum of three marks: A message/document is hashed (1) then a message digest is generated (1) which is then encrypted with the sender's private key (1) Accept any other appropriate wording/phrasing.		3

Question number	Answer	Additional Guidance	Mark
12	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks. • The (cryptographic) hash of block 2 will not match the publicly distributed ledger value recorded (1) because hashing different data produces a different hash value (1) • The (cryptographic) hash of block 2 will have changed (1) because each block contains the hash of the previous block (1) • Changing one block invalidates the following chain (1) because all succeeding blocks incorporate data from preceding blocks (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
13	Award up to two marks for a linked definition that makes reference to: • Different objects can respond to the same method in different ways (1) based on their specific class type (1) • A derived/child class can override a method with the same name in the base/parent class (1) to exhibit different behaviour (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark									
14	Award one mark for each of the following up to a maximum of four marks: • +getStrength(), +setStrength() and +display() are public methods in both Character and Warrior classes (1) • #name and #strength are protected attributes of Character (1) • #name and #strength are protected attributes of Warrior (1) • -status is a private attribute of Warrior only (1) Exemplar answer: <table><tr><td>Character</td></tr><tr><td>#name</td></tr><tr><td>#strength</td></tr><tr><td>+getStrength() +setStrength() +display()</td></tr></table>  <table><tr><td>Warrior</td></tr><tr><td>#name</td></tr><tr><td>#strength</td></tr><tr><td>-status</td></tr><tr><td>+getStrength() +setStrength() +display()</td></tr></table>	Character	#name	#strength	+getStrength() +setStrength() +display()	Warrior	#name	#strength	-status	+getStrength() +setStrength() +display()	Allow transcription errors as long as intent is clear. Do not award incorrect use of +, -, (), # Award only attributes in centre cells and only methods in lower cells. Award any order within the correct cell. Ignore case and minor transcription errors.	4
Character												
#name												
#strength												
+getStrength() +setStrength() +display()												
Warrior												
#name												
#strength												
-status												
+getStrength() +setStrength() +display()												

Question number	Answer	Additional Guidance	Mark																																																																																																																								
15	Award one mark for each of the following up to a maximum of six marks: • Initialisation of R0 to R3 (1) • R4 calculation to determine if R1 is odd (1) • R5 calculation to multiply R0 and update answer in R4 (1) • R2 calculation to increment R2 and integer divide R1 by 2 (1) • All final register values correctly calculated (1) • Purpose of algorithm given as to multiply the values given in R0 and R1 together and place the answer in R3 (1) Exemplar answer: <table><tr><th>R0</th><th>R1</th><th>R2</th><th>R3</th><th>R4</th><th>R5</th></tr><tr><td>6</td><td>5</td><td>0</td><td>0</td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>5</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>1</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td>6</td></tr><tr><td></td><td></td><td></td><td>6</td><td></td><td></td></tr><tr><td></td><td></td><td>1</td><td></td><td></td><td></td></tr><tr><td></td><td>2</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>2</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>0</td><td></td></tr><tr><td></td><td></td><td>2</td><td></td><td></td><td></td></tr><tr><td></td><td>1</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>1</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>1</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td>24</td></tr><tr><td></td><td></td><td></td><td>30</td><td></td><td></td></tr><tr><td></td><td></td><td>3</td><td></td><td></td><td></td></tr><tr><td></td><td>0</td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td>0</td><td>3</td><td>30</td><td>1</td><td>24</td></tr><tr><td colspan="2">Program function</td><td colspan="4">Multiply the values in R0 and R1 and place result in R3</td></tr></table>	R0	R1	R2	R3	R4	R5	6	5	0	0							5						1							6				6					1					2									2						0				2					1									1						1							24				30					3					0					6	0	3	30	1	24	Program function		Multiply the values in R0 and R1 and place result in R3				Mark points in the trace table are colour coded: BP1 BP2 BP3 BP4 BP5 BP6	6
R0	R1	R2	R3	R4	R5																																																																																																																						
6	5	0	0																																																																																																																								
				5																																																																																																																							
				1																																																																																																																							
					6																																																																																																																						
			6																																																																																																																								
		1																																																																																																																									
	2																																																																																																																										
				2																																																																																																																							
				0																																																																																																																							
		2																																																																																																																									
	1																																																																																																																										
				1																																																																																																																							
				1																																																																																																																							
					24																																																																																																																						
			30																																																																																																																								
		3																																																																																																																									
	0																																																																																																																										
6	0	3	30	1	24																																																																																																																						
Program function		Multiply the values in R0 and R1 and place result in R3																																																																																																																									

Question number	Answer	Additional Guidance	Mark
16	Award one mark for: • Availability (1)		1

Question number	Answer	Additional Guidance	Mark
17	Award one mark for: • DNS/Domain Name System (1)		1

Question number	Answer	Additional Guidance	Mark
18	The only correct answer is D Answer A, B and C have all been transmitted using even parity.	The correct answer is: D 1010 0100	1

Question number	Answer	Additional Guidance	Mark
19	Award one mark for each appropriate point given in the correct order, up to a maximum of two marks: • Electrical signals from a controller are received (1) then converted into physical outputs (1) • A control signal operates an energy source (1) to perform a mechanical operation (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
20	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks. • A request to resend (a corrupted packet) can be sent to the sender (1) because the source IP address can be set as the destination address (1) • Firewall can block the packet being transmitted/received (1) because packets from specific source IP addresses can be blocked (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
21	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks. • RSA encryption would take too long to crack (using brute force) (1) because factoring a very large number into its two prime factors has no polynomial time algorithm (1) • Two very large primes multiplied together form a secure key (1) because factorising the resultant key has no polynomial time algorithm (1) • The product of two primes p and q ($n = pq$) can be computed in polynomial time (1) whereas factoring the product n back into p and q takes exponential time (1) Accept any other appropriate response.	Accept “reasonable time” as an alternative to polynomial time.	2

Question number	Answer	Additional Guidance	Mark
22	Award one mark for: • Adding/removing an edge takes $O(1)$ time / is faster (1) • Determining if two nodes are connected takes $O(1)$ time / is faster (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
23(a)	Award one mark for each appropriate point given in the correct order, up to a maximum of four marks: Purple is first compared to the root/Pink(1) it is greater (than the root/Pink so the right subtree is searched) (1) it is less than Red (so the left subtree is searched) (1) and is then compared to Purple so a (match is found) (1) Accept any other appropriate response.		4

Question number	Answer	Additional Guidance	Mark
23(b)	Award one mark for: Pre-order (traversal) (1)		1

Question number	Answer	Additional Guidance	Mark
24	Award one mark for each appropriate point in a linked response up to a maximum of four marks: The function count() is called with HeadPointer as the argument (1) 0 is returned if the base case pointer equals None (1), otherwise the function returns 1 plus (1) the recursive call to the current node's next pointer (1) Accept any other appropriate response.	Allow responses that use relevant code such as: <pre>def count(node): if node == None: return 0 else: return 1 + count(node.next)</pre>	4

Question number	Answer	Additional Guidance	Mark
25	The only correct answer is D. A is incorrect because it describes logarithmic complexity B is incorrect because it describes constant complexity C is incorrect because it describes exponential complexity	The correct answer is: D – Its run time grows in direct proportion to the number of items to process is the correct answer.	1

Question number	Answer	Additional Guidance	Mark						
26	Award one mark for each of the following up to a maximum of two marks: <table><thead><tr><th>Operation</th><th>Average case</th></tr></thead><tbody><tr><td>Binary search</td><td>$O(\log n)$ (1)</td></tr><tr><td>Merge sort</td><td>$O(n \log n)$ (1)</td></tr></tbody></table>	Operation	Average case	Binary search	$O(\log n)$ (1)	Merge sort	$O(n \log n)$ (1)		2
Operation	Average case								
Binary search	$O(\log n)$ (1)								
Merge sort	$O(n \log n)$ (1)								

Question number	Answer	Additional Guidance	Mark
27	Award one mark for each appropriate point in a linked response up to a maximum of two marks: Values less than the pivot are placed to the left of it (1) and values greater than the pivot are placed to the right of it (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
28	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks. • Heuristic rules can be applied (1) so that a solution that runs in polynomial time can be produced (1) • The problem can be reframed/relaxed using heuristics (1) to allow an approximation to an optimal solution to be found (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
29	Award four marks for a fully correct expression: • $A.B + B.\overline{D} + \overline{A}.\overline{B}.D$ (4) OR Award one mark for any of the following up to a maximum of three marks for working: • $A.B$ (1) • $B.\overline{D}$ (1) • $\overline{A}.\overline{B}.D$ (1) • Each product term connected with a logical or (1)	Accept AB instead of $A.B$ Do not accept $\overline{AB}$ instead of $\overline{A}.\overline{B}$ Do not accept answers that are not given in the correct notation.	4

Question number	Answer	Additional Guidance	Mark
30	Award one mark for each of the following up to a maximum of three marks: • Correct application of De Morgan's Laws (1) • Correct application of Distributive Law (1) • Fully correct expression (1) Exemplar full method: $\bar{X}.\bar{Y} + Y.\bar{X} \text{ or } \overline{\bar{X}.\bar{Y}} + Y.\bar{X}$ $X.Y + Y.\bar{X}$ $Y.(X + \bar{X})$ $Y.1$ Y	$\bar{X}.\bar{Y} + Y.\bar{X}$ or $\overline{\bar{X}.\bar{Y}} + Y.\bar{X}$ De Morgan's Law $X.Y + Y.\bar{X}$ Double negation Law $Y.(X + \bar{X})$ Distributive Law $Y.1$ Annulment Law Y Identity Law	3

Question number	Answer	Additional Guidance	Mark																																							
31	Award one mark for each of the following up to a maximum of six marks: • Rows A, B and C (1) • Row D (1) • Rows E and F (1) • Rows G and H (1) • Shortest path ACDFGH (1) • Shortest distance 20 (1) Exemplar response: <table><tr><th>Node</th><th>Distance to node</th><th>Path to Node</th></tr><tr><td>A</td><td>0</td><td>-</td></tr><tr><td>B</td><td>5</td><td>A</td></tr><tr><td>C</td><td>7</td><td>A</td></tr><tr><td>D</td><td>12</td><td>B</td></tr><tr><td></td><td>11</td><td>C</td></tr><tr><td>E</td><td>12</td><td>B</td></tr><tr><td>F</td><td>14</td><td>D</td></tr><tr><td>G</td><td>18</td><td>E</td></tr><tr><td></td><td>16</td><td>F</td></tr><tr><td>H</td><td>20</td><td>G</td></tr><tr><td>Distance</td><td>20</td><td></td></tr><tr><td>Shortest path</td><td>ACDFGH</td><td></td></tr></table>	Node	Distance to node	Path to Node	A	0	-	B	5	A	C	7	A	D	12	B		11	C	E	12	B	F	14	D	G	18	E		16	F	H	20	G	Distance	20		Shortest path	ACDFGH			6
Node	Distance to node	Path to Node																																								
A	0	-																																								
B	5	A																																								
C	7	A																																								
D	12	B																																								
	11	C																																								
E	12	B																																								
F	14	D																																								
G	18	E																																								
	16	F																																								
H	20	G																																								
Distance	20																																									
Shortest path	ACDFGH																																									

Pearson Edexcel International Advanced Level

Sample assessment material

Paper
reference

WCP03/01

Computer Science

International Advanced Level

Unit 3: Advanced Principles of Computer Science

Assembly Language Instruction Set

Resource Booklet

Do not return this Booklet with the question paper.

S89027A

©2025 Pearson Education Ltd.
1/



Arithmetic and logical operations

ADD Rd, Rn, <operand>	Add the value in <operand> to the value in register Rn and store the result in register Rd.
SUB Rd, Rn, <operand>	Subtract the value in <operand> from the value in register Rn and store the result in register Rd.
AND Rd, Rn, <operand>	Perform a bitwise logical AND between the values in register Rn and <operand>, and store the result in register Rd.
ORR Rd, Rn, <operand>	Perform a bitwise logical OR between the values in register Rn and <operand>, and store the result in register Rd.
EOR Rd, Rn, <operand>	Perform a bitwise logical XOR between the values in register Rn and <operand>, and store the result in register Rd.
MVN Rd, <operand>	Performs a bitwise logical NOT to the value in <operand>, and store the result in register Rd.
LSL Rd, Rn, <operand>	Logically shift the value stored in register Rn to the left by the number of bits specified by <operand> and store the result in register Rd.
LSR Rd, Rn, <operand>	Logically shift the value stored in register Rn to the right by the number of bits specified by <operand> and store the result in register Rd.

Comparison and branch instructions

CMP Rn, <operand>	Compare the value in <operand> with the value stored in register Rn and set status bits based on the outcome.
B <label>	Unconditionally branch to the address at position <label>.
B<condition> <label>	Conditionally branch to the instruction at position <label> when the last CMP instruction result matches the specified <condition>.

Data movement and other instructions

MOV Rd, <operand>	Move the value in <operand> into register Rd.
LDR Rd, <address>	Load the contents of the memory location <address> into register Rd.
STR Rd, <address>	Store the contents of Rd to the memory location <address>.
HALT	Halt program execution.

Syntax notes

<operand>	References either #x decimal value or Rn register n
<condition>	EQ equal to, NE not equal to, GT greater than, LT less than
<label>:	A label definition is formed by an identifier followed by a colon
Rd, Rn	General purpose registers R0 to R12

Pearson Edexcel International Advanced Level

Sample assessment material

Time 3 hours

Paper
reference

WCP04/01

Computer Science

International Advanced Level

Unit 4: Advanced Practical Programming and Problem-solving

You must have:

- a computer workstation with appropriate programming language code editing software and tools, including an IDE that you are familiar with that shows line numbers
- a 'STUDENT CODING' folder containing code and data files
- digital copy of the program language subset (PLS)

Instructions

- Answer **all** questions on your computer.
- Save the new or amended code in the 'COMPLETED CODING' folder using the name given in the question.
- Do **not** overwrite the original code and data files provided to you.
- You must **not** use the internet at any time during the examination.

Information

- The total mark for this paper is 80.
- The marks for **each** question are shown in brackets
– *use this as a guide as to how much time to spend on each question.*
- The 'STUDENT CODING' folder in your user area includes all the code and data files you need.

Advice

- Read each question carefully before you start to answer it.
- Save your work regularly.
- Check your answers and work if you have time at the end.
- Try to answer every question.

S85797A

©2026 Pearson Education Ltd.
1/1/1/1/1/1/1/1



Answer ALL questions.

Suggested time: 15 minutes

- 1** A program is intended to implement searching and validation with regular expressions.

Open file **Q01.py**

Amend the code to:

- fix the syntax error on original line 5
`import`
- fix the syntax error on original line 15
`"4. UpDown 15 GB 12 months 120 minutes 50 texts £4.44
per month"`
- fix the logic error on original line 25
`regex = r" "`
- fix the `AttributeError` and the `NameError` on original line 29
`match = re.splot(regextra, offer)`
- fix the logic error on original line 36, which should find all the offers with an unlimited number of minutes
`regex = r"minutes"`
- fix the logic error on original line 45
`regex = r"^2"`
- fix the logic error on original line 54
`regex = r"[letter][number][letter]"`
- fix the logic error on original line 65 to validate the values for DD in DD-MM-YY.
`regex = r"^(0$"`

The valid values for DD are:

- the digit zero followed by a digit one to nine:
 - 01, 02, 03, 04, 05, 06, 07, 08, 09
- the digit one or two followed by a digit zero to nine:
 - 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
 - 20, 21, 22, 23, 24, 25, 26, 27 28, 29
- the digit three followed by the digit zero or one:
 - 30, 31

Execute the code to test its functionality.

Do **not** change the intended functionality of the provided code.

Do **not** add any additional functionality.

Save your amended code as **Q01FINISHED.py**

(Total for Question 1 = 12 marks)

Suggested time: 20 minutes

- 2 A program uses the functional programming techniques of pure functions and lazy evaluation.

Pure function

A pure function only changes states, variables and data that are created inside the function and always returns a result. Pure functions have no side effects, such as changing global variables.

Lazy evaluation

Lazy evaluation delays evaluation of an expression until its value is needed.

The program

The `add_member()` function adds a member to a new list.

The member key is the first item in each record of the list.

The user enters the birth year and the birth month.

The key is calculated by adding the birth year to the birth year modulo the birth month. Key collisions are ignored.

The `lazy_fives()` function produces a sequence of integers evenly divisible by five, using lazy evaluation.

The output is from a fully-functional program when the user enters a birth month of 6 and a birth year of 2005.

Output

```
Enter your birth month (1 is January): 6
Enter your birth year in four digits: 2005

Original members before add_member call
[[1996, 'Adams', 'Amanda', 1, 1996],
 [2002, 'Brown', 'Brenda', 6, 2000],
 [2000, 'Collins', 'Christopher', 3, 1999]]

Original members after add_member call
[[1996, 'Adams', 'Amanda', 1, 1996],
 [2002, 'Brown', 'Brenda', 6, 2000],
 [2000, 'Collins', 'Christopher', 3, 1999]]

New list of members after add_member call
[[1996, 'Adams', 'Amanda', 1, 1996],
 [2002, 'Brown', 'Brenda', 6, 2000],
 [2000, 'Collins', 'Christopher', 3, 1999],
 [2006, 'Wilson', 'Wilma', 6, 2005]]

0 5 10
```

Open file **Q02.py**

Amend the code to:

- implement `add_member()` as a pure function
- implement `lazy_fives()` so that it can be evaluated lazily
- implement the main program to call the functions.

Use the variables provided.

Execute the code to test its functionality.

Do **not** change the intended functionality of the provided code.

Do **not** add any additional functionality.

Save your amended code as **Q02FINISHED.py**

(Total for Question 2 = 13 marks)

Suggested time: 45 minutes

- 3 A department is modelled using object-oriented programming. The department has employees. Each employee is a person.

The output is from a fully-functional program.

Output

```
===== Create and show a person =====  
Person 1  
Ansel Alice  
===== Create and show employees =====  
Employee 1  
Banks Barbara 30/04/2015  
Employee 2  
Cash Connie 29/04/2020  
===== Create and show the department =====  
The Department  
Banks Barbara 30/04/2015  
Cash Connie 29/04/2020
```

Open file **Q03.py**

Amend the code to:

- create and show a person, by adding code to the Person class and the main program
- create and show employees, by adding code to the Employee class and the main program
- create and show the department, by adding code to the Department class and the main program.

Use the classes, attributes, methods and variables provided.

Execute the code to test its functionality.

Do **not** change the intended functionality of the provided code.

Do **not** add any additional functionality.

Save your amended code as **Q03FINISHED.py**

(Total for Question 3 = 17 marks)

Suggested time: 45 minutes

- 4 The names of eight foods are stored in a list of records. The names must be inserted into a binary search tree.

Binary search tree

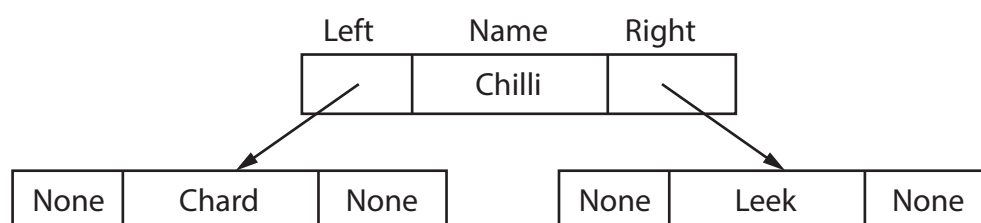
Each node, in the binary search tree, contains a name and two pointer values.

The binary search tree is ordered:

- a left pointer points to a node with a name earlier in the alphabet
- a right pointer points to a node with a name later in the alphabet
- a pointer value of None represents no pointer.

The binary search tree shows the initial state of the tree given in the program code file.

Binary search tree



The output is from a fully-functional program.

Output

```
===== Building tree =====
===== Tree complete =====
===== Post Order Traversal =====
Carrot
Beet Root
Chard
Fennel
Peas
Sweet Corn
Squash
Lettuce
Tomato
Leek
Chilli
```

Open file **Q04.py**

Amend the code to:

- create a procedure to insert a node into the binary search tree. The node is passed as a parameter
- create a new node for each food in the list of food. The program must work for any length of list
- insert each new node into the binary search tree, using the insert procedure
- create a procedure that implements a recursive post-order traversal of the binary search tree to display the food name in each node
- call the post-order traversal procedure.

Use the class and variables provided.

Execute the code to test its functionality.

Do **not** change the intended functionality of the provided code.

Do **not** add any additional functionality.

Save your amended code as **Q04FINISHED.py**

(Total for Question 4 = 18 marks)

Suggested time: 55 minutes

- 5 A program is required to calculate the scores of dives and display leader boards for a 10-metre platform synchronised diving competition.

Synchronised diving is when two people dive at the same time, side by side, from the same platform.

The competition has four rounds, with one dive per team. There are five teams. Each team has two divers.

Each dive is scored by 11 judges:

- three judges score diver 1
- three judges score diver 2
- five judges score the synchronisation.

A judge's score is a real number, between 0 and 10, in 0.5 increments.

Input

- the team number and name of the diving teams, stored in a dictionary
- header titles for the leader board, stored in a list
- a leader board, holding the team numbers, in numerical order by team, stored in a two-dimensional list
- four rounds, holding information about one dive for each team, in numerical order by team, stored in a two-dimensional list.

Round 1 input shows the information given in the program code file.

Round 1 input

Team number		Difficulty	Judges' scores diver 1			Judges' scores diver 2			Judges' scores synchronisation				
		↓											
[	[17,	2.0,	8.5,	9.0,	8.5,	9.0,	8.5,	9.5,	8.5,	9.5,	9.0,	9.0,	9.5],
	[23,	2.0,	9.0,	8.0,	9.0,	8.0,	9.0,	9.0,	8.5,	8.5,	8.0,	8.0,	8.5],
	[28,	3.3,	9.5,	9.0,	9.0,	9.0,	9.5,	9.0,	8.5,	9.5,	8.0,	8.5,	9.5],
	[45,	3.0,	9.5,	9.5,	9.5,	8.0,	9.5,	8.5,	8.5,	9.0,	8.0,	9.5,	9.0],
	[83,	3.0,	9.0,	9.5,	8.0,	8.0,	8.0,	9.0,	9.5,	9.5,	8.5,	9.0,	9.5]
		]											

Processes

- The raw score for a dive is calculated as the sum of three median scores:
 - median of diver 1's three scores
 - median of diver 2's three scores
 - median three scores of the five scores for synchronisation.

For team 17 in **round 1 input**, the calculation of the raw score is:

$$8.5 + 9.0 + (9.0 + 9.0 + 9.5) = 45.0$$

- The round score for a dive is calculated to two decimal places as:
 - raw score $\times$ scale adjustment of 0.6 $\times$ difficulty

For team 17 in **round 1 input**, the calculation of the round score is:

$$45.0 \times 0.6 \times 2.0 = 54.00$$

- The total score after each round is the sum of the round scores so far.

Output

- Four leader boards, showing the round scores and the total score for the team.
The leader board is sorted in descending order by total score.

Round 1 output shows the leader board at the end of round 1, from a fully-functional program.

Round 1 output

Team	Name	Round 1	Round 2	Round 3	Round 4	Total
28	Warriors	88.11	–	–	–	88.11
83	Chargers	81.00	–	–	–	81.00
45	Orcas	80.10	–	–	–	80.10
17	Pirates	54.00	–	–	–	54.00
23	Hawks	51.60	–	–	–	51.60

Round 4 output shows the leader board at the end of round 4, from a fully-functional program.

Round 4 output

Team	Name	Round 1	Round 2	Round 3	Round 4	Total
28	Warriors	88.11	88.32	87.72	44.37	308.52
83	Chargers	81.00	72.24	84.48	49.59	286.41
45	Orcas	80.10	63.36	54.81	71.28	270.45
17	Pirates	54.00	70.47	56.10	82.77	263.34
23	Hawks	51.60	76.50	73.71	58.74	260.55

Open file **Q05.py**

Write the code to:

- calculate the scores after each round of dives
- display the leader board after each round of dives
- ensure the original dive data is not changed by the processing.

Use best practice techniques to:

- design effective and efficient code
- write clean code
- ensure the solution functions as intended.

Use the constants and variables provided.

Execute the code to test its functionality.

Do **not** change the intended functionality of the provided code.

Do **not** add any additional functionality.

Save your amended code as **Q05FINISHED.py**

(Total for Question 5 = 20 marks)

TOTAL FOR PAPER = 80 MARKS

International Advanced Level in Computer Science

Unit 4: Advanced Practical Programming and Problem- solving

Mark Scheme

General marking guidance

- All candidates must receive the same treatment. Examiners must mark the first candidate in exactly the same way as they mark the last.
- Mark schemes should be applied positively. Candidates must be rewarded for what they have shown they can do rather than penalised for omissions.
- Examiners should mark according to the mark scheme not according to their perception of where the grade boundaries may lie.
- There is no ceiling on achievement. All marks on the mark scheme should be used appropriately.
- All the marks on the mark scheme are designed to be awarded. Examiners should always award full marks if deserved, i.e. if the answer matches the mark scheme. Examiners should also be prepared to award zero marks if the candidate's response is not worthy of credit according to the mark scheme.
- Where some judgement is required, mark schemes will provide the principles by which marks will be awarded and exemplification may be limited.
- When examiners are in doubt regarding the application of the mark scheme to a candidate's response, the team leader must be consulted.

This mark scheme includes colour coded text. Centres must ensure that all hard copies are printed in full colour so that assessment guidance is accurately conveyed.

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
1	1.1	5	The regular expression library is imported <code>import re</code>	Allow any method	12
	1.2	15	Closing square bracket] added to close declaration and initialisation of list "4. UpDown 15 GB 12 months 120 minutes 50 texts £4.44 per month"]		
	1.3	25	The logic error is fixed by adding the missing string between quotes <code>regex = r"no contract"</code>		
	1.4	29	The attribute error is fixed by correcting splot to search <code>match = re.search(regex, offer)</code>	Award red only	
	1.5	29	The name error is fixed by correcting regextra to reg <code>match = re.search(regex, offer)</code>	Award red only	
	1.6	36	The logic error is corrected by supplying the missing text for the expression to the left of minutes <code>regex = r"[uU]nlimited minutes"</code>	Allow any method [unlimited Unlimited]	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
1 (continued)	1.7	45	The logic error is corrected by supplying the additional parts of the expression and the curly braces regex = r"^[a-zA-Z]{2}"	Allow "[a-zA-Z][a-zA-Z]"	12
	1.8	54	The logic error is corrected by replacing the contents of the regex expression for digits regex = r"^[a-zA-Z]{2}[0-9]{3}[a-zA-Z]{2}\$"	- Award red only - One mark for the three digits, wherever they appear	
	1.9	54	The logic error is corrected by replacing the contents of the regex expression for digits regex = r"^[a-zA-Z]{2}[0-9]{3}[a-zA-Z]{2}\$"	- Award red only - One mark for one instance of the two letters, wherever it appears	
	1.10	65	The logic error is corrected by adding the missing parts of the day number regex = r"^(0[1-9] [12][0-9] 3[01])\$"	- One mark for each of the following up to a maximum of two marks: - Days: 01-09 - Days: 30, 31 - Days: 10-19, 20-29	
	1.11	65			
	1.12	65	The logic error is corrected by adding the missing parts of the day number for all three cases (purple, turquoise, red) regex = r"^(0[1-9] [12][0-9] 3[01])\$"	- Completely accurate expression including use of ^, , and \$ - Sub-expressions between the ORs could be in different orders	

Mark points	Guidance
MP1.10	Do not award either mark, where a sequence of all the numbers 01–31 is supplied using a simple OR (), e.g.:
MP1.11	(01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31)

```

1  # -----
2  # Import libraries
3  # -----
4  # =====> Fix the syntax error
5  import re
6
7  # -----
8  # Global variables
9  # -----
10 # =====> Fix the syntax error
11 offers = [
12     "1. Seven mobile 24 months 3 GB unlimited minutes Unlimited texts £5.00
13     per month",
14     "2. Able network no contract 1 GB 200 minutes Unlimited texts £3.95 per
15     month",
16     "3. MobileTalk 1 month 30 GB Unlimited minutes unlimited texts £6.95 per
17     month",
18     "4. UpDown 15 GB 12 months 120 minutes 50 texts £4.44 per month"]
19
20 # -----
21 # Main program
22 # -----
23 # Find all offers with no contract
24 print("----- No contract -----")
25 # =====> Fix the logic error
26 regex = r"no contract"
27 for offer in offers:
28     match = re.search(regex, offer)
29     if (match is not None):
30         print(offer)
31
32 # Find all offers with unlimited minutes
33 print("----- Unlimited minutes -----")
34 # =====> Fix the logic error
35 regex = r"[uU]nlimited minutes"      # regex = r"[unlimited|Unlimited] minutes"
36 for offer in offers:
37     match = re.search(regex, offer)
38     if (match is not None):
39         print(offer)

```

```

40 # Find all keys starting with two alphabetic characters
41 print("----- Keys starting with two alphabetic characters -----")
42 # =====> Fix the logic error
43 regex = r"^[a-zA-Z]{2}" # regex = r"^[a-zA-Z][a-zA-z]"
44 for key in keys:
45     match = re.search(regex, key)
46     if (match is not None):
47         print(key)
48
49 # Find all keys ending with three non-zero digits
50 print("----- Keys ending with three non-zero digits -----")
51 # =====> Fix the logic error
52 regex = r"[1-9]{3}$"
53 for key in keys:
54     match = re.search(regex, key)
55     if (match is not None):
56         print(key)
57
58 # Find all keys with the pattern LLNNNLL
59 print("----- Keys with pattern LLNNNLL")
60 # =====> Fix the logic error
61 regex = r"^[a-zA-Z]{2}[0-9]{3}[a-zA-Z]{2}$"
62 for key in keys:
63     match = re.search(regex, key)
64     if (match is not None):
65         print(key)
66
67 # Validate the day part of a date
68 # Two digits exactly
69 # Leading zeros required
70 print("----- Day numbers with format of DD -----")
71 # =====> Fix the logic error
72 regex = r"^(3[01]|[12][0-9]|0[1-9])$"
73 in_string = input("Enter a day number: ")
74 while (in_string != ""):
75     match = re.search(regex, in_string)
76     if (match is not None):
77         print(in_string, "Pass")
78     else:
79         print(in_string, "Fail")
80     in_string = input("Enter: ")
81

```

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
2			add_member Function		13
	2.1	18	Parameter for last name, first name, year and month are added to definition header <code>def add_member(p_members_list, p_last, p_first, p_month, p_year)</code>	• Award red only • Allow any order • Ignore any other changes to the line	
	2.2	21	Calculation of new key is implemented using the input parameters <code>new_key = p_year + (p_year % p_month)</code>		
	2.3	24	A new record is created for the table <code>new_record = [new_key, p_last, p_first, p_month, p_year]</code>	• Award red only • Allow any order for fields	
	2.4	24	The order of the fields in the records is correct <code>new_record = [new_key, p_last, p_first, p_month, p_year]</code>	• Award field order only • Ignore names	
	2.5	30	The new record is added to the original member list parameter <code>new_members_list = p_members_list + [new_record]</code>	• Award red only • New list construction	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
2 (continued)	2.6	30	A new 2D list (table) is generated <code>new_members_list = p_members_list + [new_record]</code>	• Award red only • Allow any meaningful identifier for the new list • Assignment only	13
	2.7	33	The new list is returned <code>return(new_members_list)</code>	• Ignore brackets • Variable name must match variable created by appending	
	lazy_fives Function				
	2.8	38	The line is completed to generate numbers from zero to p_num <code>num in range(p_num):</code>		
	2.9	42	The line is completed to determine if an integer is evenly divisible by five <code>if (num % 5 == 0):</code>	• Award red only • Relational test only	
	2.10	45	A line is added to ensure lazy evaluation <code>yield(num)</code>		

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
2 (continued)			Main Program		13
	2.11	61	The add_member pure function is called with arguments in the order to match the parameters <code>newList = add_member(members, "Wilson", "Wilma", birth_month, birth_year)</code>	Order of arguments must match order of parameters for <code>add_member()</code> [appx. Line 18]	
	2.12	71	The lazy_fives function is assigned to a callable name, i.e. the generator function <code>gen_function = lazy_fives(30)</code>	- Assignment only - Allow any function name	
	2.13	74	The next available value available from the generator functions is retrieved by using <code>next(callable name)</code> <code>print(next(gen_function), next(gen_function), next(gen_function))</code>	Allow any method <pre>for count in range(3): print (next(gen_function), end = " ") print(str(next(gen_function)) + " " + str(next(gen_function)) + " " + str(next(gen_function))) print(".join([str(next(gen_function)) for i in range(3)])])</pre>	

```

1  from pprint import pprint          # For debugging only
2
3  # -----
4  # Global variables
5  # -----
6  # Key, last name, first name, month, year
7  members = [[1996, "Adams", "Amanda", 1, 1996],
8              [2002, "Brown", "Brenda", 6, 2000],
9              [2000, "Collins", "Christopher", 3, 1999]]
10
11 birth_year = 0
12 birth_month = 0
13
14 # -----
15 # Subprograms
16 # -----
17 # =====> Complete the definition for the pure function
18 def add_member(p_members_list, p_last, p_first, p_month, p_year):
19
20     # =====> Complete the calculation to produce a new key
21     new_key = p_year + (p_year % p_month)
22
23     # =====> Complete the line to create a new record for the member
24     new_record = [new_key, p_last, p_first, p_month, p_year]
25
26     # =====> Complete the line to create a new list by adding the
27     #             new record to the end of the member list.
28     #             Changing the values in the input list would produce
29     #             a side effect
30     new_members_list = p_members_list + [new_record]
31
32     # =====> Complete the line to return the new list
33     return(new_members_list)
34

```

```

35 def lazy_fives(p_num):
36
37     # =====> Complete the line to generate integers from zero to p_num
38     for num in range(p_num):
39
40         # =====> Complete the line to determine if an integer
41         #             is evenly divisible by five
42         if (num % 5 == 0):
43
44             # =====> Add a line to ensure lazy evaluation of the
45             #             function
46             yield(num)                # Causes the evaluation to be
47             lazy
48
49 # -----
50 # Main program
51 # -----
52 birth_month = int(input("Enter your birth month (1 is January): "))
53 birth_year = int(input("Enter your birth year in four digits: "))
54
55 # Debugging only - for highlighting side effects
56 print("\nOriginal members before add_member call")
57 pprint(members)
58
59 # =====> Complete the line to call the pure function to add a
60 #             new member with a last name of Wilson, a first name
61 #             of Wilma, a birth month as entered, and a birth year
62 #             as entered
63 newList = add_member(members, "Wilson", "Wilma", birth_month, birth_year
64 )
65
66 # Debugging only - For highlighting side effects
67 print("\nOriginal members after add_member call")
68 pprint(members)
69 print("\nNew list of members after add_member call")
70 pprint(newList)
71 print("\n")
72
73 # =====> Complete the line to assign the output of lazyFives to a
74 #             generator function
75 gen_function = lazy_fives(30)
76
77 # =====> Add a line to display the first three values generated by the
78 #             call to lazyFives, all on a single line.
79 print(next(gen_function), next(gen_function), next(gen_function))

```

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
3			Person Class		17
	3.1	12	The last name is returned from the get_last() function <code>return(self.__last_name)</code>		
	3.2	16	The first name returned from get_first() function <code>return(self.__first_name)</code>		
	3.3	21	A string is constructed consisting of last name, space, first name constructed in the get_info() method <code>person_info = self.__last_name + " " + self.__first_name</code>	• Allow any method • Allow any order of fields	
	3.4	22	The string value is returned from get_info() method <code>return(person_info)</code>	• Allow construction of string inside return	
			Employee Class		
	3.5	26	Person is added to class definition header <code>class Employee(Person):</code>		

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
3 (continued)	3.6	32	The base class constructor is called <code>super().__init__(p_last, p_first)</code>	- Two arguments required - Ignore order of arguments - Do not award any other method, as this MP is not for functionality	17
	3.7	38	The get_info method in the base class is called to get last names and first names <code>employee_string = super().get_info() + " " + self.__start_date</code>	- Award red only - Award concatenation by any method - Do not award any other method, as this MP is not for functionality	
	3.8	38	The private attribute __start_date is appended to output of get_info() separated by a space <code>employee_string = super().get_info() + " " + self.__start_date</code>	- Award red only - Award concatenation by any method	
	Department Class				
	3.9	46	None is changed to [] <code>self.__emp_list = []</code>	Ignore any attempt to dimension the list	
	3.10	51	Public method add_employee header is implemented <code>def add_employee(self, p_employee):</code>	Allow any name for second parameter	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
3 (continued)	3.11	52	The input parameter is appended to list <code>self.__emp_list.append(p_employee)</code>		17
			Main Program		
	3.12	64	Person class is used to create an instance <code>person1 = Person("Ansel", "Alice")</code>	• Allow any variable name • Allow reversal of arguments	
	3.13	66	The <code>get_info()</code> method of the instance is called <code>print(person1.get_info())</code>	• Output to display required	
	3.14	71 77	Instantiation is implemented using Employee class constructor <code>employee1 = Employee("Banks", "Barbara", "30/04/2015")</code> <code>employee2 = Employee("Cash", "Connie", "29/04/2020")</code>	• At least one instance • Allow any variable name • Allow reversal of arguments	
	3.15	73 79	The <code>get_info()</code> method in the instantiated object is called to retrieve the employee info <code>print(employee1.get_info())</code> <code>print(employee2.get_info())</code>	• At least one instance • Output to display required	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
3 (continued)	3.16	85 86	The add_employee method of the Department instance is called the_dept.add_employee (employee1) the_dept.add_employee (employee2)	- At least one instance - Ignore functionality of method add_employee()	17
	3.17	88	All the employees are accurately displayed by calling the show method of department the_dept.show_dept()	- Matches output given in paper - Order of employees may be reversed, depending on order added to the employee list	

Anticipated output

```
===== Create and show a person =====  
Person 1  
Ansel Alice  
===== Create and show employees =====  
Employee 1  
Banks Barbara 30/04/2015  
Employee 2  
Cash Connie 29/04/2020  
===== Create and show the department =====  
The Department  
Banks Barbara 30/04/2015  
Cash Connie 29/04/2020
```

```

1  # -----
2  # Classes
3  # -----
4  class Person:
5      def __init__(self, p_last, p_first):
6          # Private attributes
7          self.__last_name = p_last
8          self.__first_name = p_first
9
10     def get_last(self):
11         # =====> Complete the getter method
12         return(self.__last_name)
13
14     def get_first(self):
15         # =====> Complete the getter method
16         return(self.__first_name)
17
18     def get_info(self):
19         # =====> Complete the get_info() method to return a string containing
20         #         __last_name and __first_name separated by a single space.
21         person_info = self.__last_name + " " + self.__first_name
22         return(person_info)
23
24     # -----
25     # =====> Amend Employee class to inherit Person class
26     class Employee(Person):
27         def __init__(self, p_last, p_first, p_startdate):
28             # Private attributes
29             self.__start_date = p_startdate
30
31             # =====> Amend the Employee constructor to call the Person constructor
32             super().__init__(p_last, p_first)
33
34             # =====> Complete the get_info() method to extend the results from
35             #         the Person get_info() method to append a single space and
36             #         the Employee attribute __start_date and return the result
37         def get_info(self):
38             employee_string = super().get_info() + " " + self.__start_date
39             return(employee_string)
40

```

```

41 # -----
42 class Department:
43     def __init__(self):
44         # Private attributes
45         # =====> Amend the line to assign an empty list to __emp_list
46         self.__emp_list = []
47
48         # =====> Add a public method named add_employee that takes
49         #         an Employee object as a parameter and adds it to the
50         #         list of employees, but returns no value
51     def add_employee(self, p_employee):
52         self.__emp_list.append(p_employee)
53
54     def show_dept(self):
55         for employee in self.__emp_list:
56             print(employee.get_info())
57
58 # -----
59 # Main program
60 # -----
61 print("="*10 + " Create and show a person " + "="*10)
62 print("Person 1")
63 # =====> Create a Person object named person1 with values: "Ansel", "Alice"
64 person1 = Person("Ansel", "Alice")
65 # =====> Call the get_info() method to display person1.
66 print(person1.get_info())
67
68 print("="*10 + " Create and show employees " + "="*10)
69 print("Employee 1")
70 # =====> Create an Employee object named employee1 with values: "Banks",
71 #         "Barbara", "30/04/2015"
72 employee1 = Employee("Banks", "Barbara", "30/04/2015")
73 # =====> Call the get_info() method to display employee1.
74 print(employee1.get_info())

```

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4			Insert procedure		18
	4.1	31	An insert subprogram header is defined def insert(p_node):	Award red only	
	4.2	31	The insert subprogram is implemented with a new node parameter def insert(p_node):	- Award red only - Award this mark where the root node is also passed as an input parameter	
	4.3	33	Manipulation of scope is demonstrated by the use of global keyword global root	Award additional input parameter, which also exemplifies understanding of scope def insert(p_node, p_root):	
	4.4	40	A while loop is used to locate insertion point while (not found):	Allow recursive solution	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4 (continued)	4.5	42	If/else is used with a relational test to determine whether the left or the right pointer is to be followed <code>if (p_node.name < pointer.name):</code>	• Accurate use of < to follow left/right with else. • Do not award two if statements • Tests could be reversed, as in if/else clauses may be swapped • Ignore inclusion of =, as there are no duplicates in the data	18
	4.6	44 51	The new node is inserted by assigning left or right pointer to the new node <code>pointer.left = p_node</code> <code>pointer.right = p_node</code>	• At least one instance	
	4.7	55	Inserting a node is implemented as a procedure, that has no return value	• Do not award this mark point if there is a return statement provided, even if mal-formed • Allow recursive solution	
	Post-order traversal				
4.8		58	The post order traversal header definition takes a node as a parameter <code>def post_order(p_node):</code>		

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4 (continued)	4.9	59 61	The node's pointer is checked to see if it is accessible, e.g is not equal to None (p_node.left is not None): (p_node.right is not None):	• Allow any method • At least one instance	18
	4.10	60 62	The tree is accessed, firstly, in the left direction, and, secondly, in the right direction post_order(p_node.left) post_order(p_node.right)	• Award red only	
	4.11	60 62	Recursive calls are used to pass in each subtree post_order(p_node.left) post_order(p_node.right)	• Ignore accuracy of traversal • At least one instance of a recursive call with an argument of a node pointer	
	4.12	63	Display of node name is done as last line of code in the procedure, i.e. on the unwinding of the recursion print(p_node.name)		
	Main program				
4.13	79	A loop is used to process every item in the food table for index in range(1, len(food)):	• Allow any method bounded by the length of the food list		

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4 (continued)	4.14	82	A new node is created, using Node class, from the food located in the list new_node = Node(food[index])	Allow creation of node inside call to insert subprogram	18
	Overall				
	4.15		A minimisation of storage space is demonstrated by using no additional copies of data structures	No copies of data structures, nodes or tree, are required	
	4.16		An understanding of scope isolation is demonstrated by minimising the use of global variables	The root is the only global variable required	
	Functionality				
	4.17		The solution: - is a reasonable attempt to solve the problem - must translate without syntax errors - must execute without runtime errors	- At least one of insert or post order traversal works correctly - Where the insert puts the nodes in the wrong order, the traversal can be awarded this mark, provided it accurately reports the state of the tree - Where the traversal is inaccurate, but the insert has put the nodes in the correct order, the insert can be awarded this mark. This may require memory inspection of the tree.	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4 (continued)	4.18		The outputs are accurate	Both the insert and the post-order traversal are accurate for the full binary tree	18

Output	Completed tree
<pre> ===== Building tree ===== ===== Tree complete ===== ===== Post Order Traversal ===== Carrot Beet Root Chard Fennel Peas Sweet Corn Squash Lettuce Tomato Leek Chilli </pre>	<pre> graph TD Chilli --> Chard Chilli --> Leek Chard --> BeetRoot[Beet Root] Chard --> Carrot Leek --> Fennel Leek --> Tomato Fennel --> Lettuce Lettuce --> Squash Lettuce --> Peas Squash --> Sweetcorn </pre>

```

1  # -----
2  # Global variables
3  # -----
4  # Seed name
5  food = ["Tomato",
6          "Lettuce",
7          "Fennel",
8          "Beet Root",
9          "Squash",
10         "Sweet Corn",
11         "Peas",
12         "Carrot"
13         ]
14
15 root = None                # Root of the binary search tree
16
17 # -----
18 # Classes
19 # -----
20 class Node(object):
21     def __init__(self, p_name):
22         self.name = p_name
23         self.right = None    # No right element
24         self.left = None    # No left element
25
26 # -----
27 # Subprograms
28 # -----
29 # =====> Create a procedure to insert a node into the binary search tree.
30 #           The node is passed as a parameter
31 def insert(p_node):
32
33     global root
34     found = False          # Loop control
35
36     # Create pointer to search the tree
37     pointer = root
38
39     # Find the place to insert the new node
40     while (not found):
41         # New record has a name to left of current pointer
42         if (p_node.name < pointer.name):
43             if (pointer.left is None):
44                 pointer.left = p_node
45                 found = True    # Have inserted to left
46             else:
47                 pointer = pointer.left
48         # New record has a name to the right of the current pointer
49         else:
50             if (pointer.right is None):
51                 pointer.right = p_node
52                 found = True    # Have inserted to right
53             else:
54                 pointer = pointer.right
55

```

```

56 # =====> Create a procedure that implements a recursive post-order traversal
57 #       of the binary search tree to display the food name in each node
58 def post_order(p_node):
59     if (p_node.left is not None):
60         post_order(p_node.left)
61     if (p_node.right is not None):
62         post_order(p_node.right)
63     print(p_node.name)
64
65 # -----
66 # Main program
67 # -----
68 # Setup up the binary search tree - Do Not amend
69 node1 = Node("Chard")
70 node2 = Node("Leek")
71 root = Node("Chilli")
72 root.right = node2
73 root.left = node1
74 # End setup
75
76 print("===== Building tree =====")
77 # =====> Create a new node for each food in the list of food
78
79 for index in range(0, len(food)):
80     # =====> Insert the new node into the binary search tree,
81     #       using the insert procedure
82     new_node = Node(food[index])
83     insert(new_node)
84 print("===== Tree complete =====")
85
86 print("===== Post Order Traversal =====")
87 # =====> Call the post-order traversal procedure
88 post_order(root)
89

```

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5	Calculating scores				20
	5.1	160 161	The median of three numbers for diver 1 and/or diver 2 is found median_diver1 = (sorted(this_record[DIVER_1:DIVER_1 + 3]))[1] median_diver2 = (sorted(this_record[DIVER_2:DIVER_2 + 4]))[1]	• Allow any method • At least one instance required • Anticipated method: pick out the fields from the record with a slice, sort the slice, index the sorted slice	
	5.2	162	The median of five numbers (synchro) is found synchro_score = (sorted(this_record[SYNCHRO:SYNCHRO + 5]))[1:4]	• Allow any method • Anticipated method: pick out the fields from the record with a slice, sort the slice, index the sorted slice	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)	5.3	163 166 169 172	The formula to calculate the total score of each dive is translated accurately <pre> raw_score = median_diver1 + median_diver2 + sum(synchro_score) raw_score = raw_score * SCALE_ADJ round_score = raw_score * this_record[DIFFICULTY] total_score = round(round_score, 2) </pre>	- Ignore resulting number if error made with medians - Allow literal for constant (SCALE_ADJ) - Allow equivalent arithmetical expressions ((Median 1 + Median 2 + Synchro) * scale adjustment * difficulty) rounded to two places ((Median 1 + Median 2 + (Synchro 1 + Synchro 2 + Synchro 3)) * scale adjustment * difficulty) rounded to two places	20
	5.4	175 142- 147	The scores are stored in a 2D data structure with team number (leader_board)	- Allow any method - May require a loop to search the leader board for the team number to find storage location	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)			Leader board		20
	5.5	114 115	The data structure, holding the scores, is sorted in reverse order, prior to display <pre>index_total = len(out_board[0]) - 1 out_board.sort(key = itemgetter(index_total), reverse = True)</pre>	Allow any method	
	5.6	99	The supplied dictionary is used to access team name based on team number <pre>out_board[out_board_row][1] = teams.get(team[0])</pre>	- Allow any method of accessing the supplied dictionary - Dictionary is supplied	
	5.7	131 132	A mechanism to handle no score symbol being displayed in the output <pre>for count in range(NUM_ROUNDS - num_dives): layout_row = layout_row + f"{NO_SCORE:^8} "</pre>	- Allow any method - Anticipate completely filling display board with NO_SCORE prior to filling with output data - Alternatively, the NO_SCORE may be added one at a time when dynamically building an f-string	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)	5.8	102 103 104	Total team score is calculated correctly from individual round scores <pre>for count in range(1, num_dives + 1): out_board[out_board_row][count + 1] = team[count] total = total + team[count]</pre>	- Allow any method - Anticipate a loop or use of sum() across a slice	20
	5.9	119	Column headers, with appropriate field widths, and separator are displayed <pre>print(f"{titles[0]:^8} {titles[1]:^10} {titles[2]:^8} {titles[3]:^8} {titles[4]:^8} {titles[5]:^8} {titles[6]:^8} print("-"*62)</pre>	- Allow <string>.format() or f-string - Do not award literal strings or attempts to manually space out the titles	
	5.10	130 133	Round scores and total score displayed with two digits after decimal point <pre>layout_row= layout_row + f"{team[2 + count]:^8.2f} " layout_row = layout_row + f"{team[6]:^8.2f}"</pre>	Formatting attempt required for this mark <pre>{^8.2f}</pre>	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)	5.11	73	Minimisation of side effects is demonstrated by keeping the team scores in an order disconnected to the display or ranking order Keeps all the scores in order of team: <code>leader_board = [[17], ...]</code> Keeps the output that is to be displayed: <code>out_board = []</code>	• Anticipated methods: • Single structure to keep the scores associated with each team that is updated with scores for each round, given in the code • Separate structure to hold the sorted scores for displaying. • This means that each time a score is updated, the whole data structure doesn't have to be searched or sorted • Allow deep copy or separately defined data structure	20
	Design				
	5.12		Modularity demonstrated by the use of subprograms <pre>def show_leader_board(p_leaders): def update_scores(p_team, p_score, p_leaders): def calc_dive_score(p_dive_data, p_leaders):</pre>	• Modularity enforced with single entry and single exit points (blocks, loops, subprograms, etc.)	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)	5.13		Parameters to reduce the need for global variables are used in subprograms <code>def show_leader_board(p_leaders):</code> <code>def update_scores(p_team, p_score, p_leaders):</code> <code>def calc_dive_score(p_dive_data, p_leaders):</code>	A solution can be written without the need for global variables, other than those provided in the student code	20
	5.14		Minimisation of storage space is demonstrated by a design that does not require copies of the data structures	A solution can be written without the need to copy any data structure.	
	Clean code				
	5.15		Clean code techniques are used: - Comments that aid understanding, but are not excessive - White space that delimits blocks of logic, but is not excessive - Consistent notation conventions and meaningful identifiers	- There must be some attempt to solve the question, with enough student-supplied code to base a decision on - For this mark, there must be at least one example of each of informative commenting, white space, consistent notation and meaningful identifiers	
	5.16		Supplied constants are used throughout	There should be no literals in code written by the student where a constant is provided	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)	5.17		Clear understanding of scope is demonstrated by parameter identifiers that indicate parameters	Do not award where global variable names are used as parameter definitions	20
	Functionality				
	5.18		The solution: - is a reasonable attempt to solve the problem - must translate without syntax errors - must execute without runtime errors	- All three required for this mark - Minimum requirement of functionality for this mark: - Every score in at least one round is processed - The calculations for a round are attempted - Formatted output is attempted	
	5.19		Some of the requirements must be met and some parts of the code must function as required	- Minimum requirement of functionality for this mark: - The round calculations - The total score calculations - Leader board sorting	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)	5.20		The outputs are accurate	Allow different column widths and alignment, if fit for purpose	20

```

1  # -----
2  # Import libraries
3  # -----
4  from operator import itemgetter      # Can use with <list>.sort() to sort display
5
6  # -----
7  # Constants
8  # -----
9  TEAM = 0                            # Index of team number
10 DIFFICULTY = 1                      # Index of degree difficulty
11 DIVER_1 = 2                        # Index of Diver 1 scores
12 DIVER_2 = 5                        # Index of Diver 2 scores
13 SYNCHRO = 8                        # Index of Synchro scores
14 NUM_ROUNDS = 4                      # Number of rounds in competition
15 NUM_TEAMS = 5                      # Number of teams
16 NO_SCORE = "-"                     # No score on leader board
17 SCALE_ADJ = 0.6                    # To bring synchronised score in line with individual
18
19 # -----
20 # Global variables
21 # -----
22 teams = {17: "Pirates",
23          23: "Hawks",
24          28: "Warriors",
25          45: "Chargers",
26          83: "Orcas"}
27
28 # Field widths for titles {:6} {:10} {:8} {:8} {:8} {:8} {:8}
29 titles = ["Team", "Name", "Round 1", "Round 2", "Round 3", "Round 4", "Total"]
30
31 # For storing the leader board
32 leader_board = [[17],
33                 [23],
34                 [28],
35                 [45],
36                 [83]]
37

```

```

38 # Team, Difficulty, 3 x Diver 1, 3 x Diver 2, 5 x Synchronisation
39 round1 = [[17, 2.0, 8.5, 9.0, 8.5, 9.0, 8.5, 9.5, 8.5, 9.5, 9.0, 9.0, 9.5],
40           [23, 2.0, 9.0, 8.0, 9.0, 8.0, 9.0, 9.0, 8.5, 8.5, 8.0, 8.0, 8.5],
41           [28, 3.3, 9.5, 9.0, 9.0, 9.0, 9.5, 9.0, 8.5, 9.5, 8.0, 8.5, 9.5],
42           [45, 3.0, 9.5, 9.5, 9.5, 8.0, 9.5, 8.5, 8.5, 9.0, 8.0, 9.5, 9.0],
43           [83, 3.0, 9.0, 9.5, 8.0, 8.0, 8.0, 9.0, 9.5, 9.5, 8.5, 9.0, 9.5]
44          ]
45
46 round2 = [[17, 2.7, 8.5, 8.5, 9.5, 8.0, 9.0, 9.0, 8.5, 9.0, 8.5, 9.5, 8.5],
47           [23, 3.0, 8.5, 9.5, 8.0, 8.5, 8.5, 8.5, 8.0, 9.0, 8.5, 9.5, 8.0],
48           [28, 3.2, 9.0, 8.5, 9.5, 8.5, 9.0, 9.5, 9.5, 9.5, 9.0, 8.0, 9.5],
49           [45, 2.8, 8.5, 8.0, 9.5, 8.0, 8.5, 8.0, 9.0, 9.0, 8.5, 8.0, 9.5],
50           [83, 2.4, 9.0, 8.0, 8.5, 9.5, 9.5, 8.5, 8.5, 8.0, 8.0, 9.5, 9.5]
51          ]
52
53 round3 = [[17, 2.2, 9.5, 8.5, 9.5, 8.5, 9.5, 8.5, 8.0, 8.0, 8.0, 8.5, 9.0],
54           [23, 2.7, 9.0, 8.5, 9.0, 9.5, 9.5, 8.5, 8.0, 9.5, 8.0, 9.5, 9.5],
55           [28, 3.4, 8.0, 9.0, 8.0, 9.5, 9.0, 9.5, 9.5, 8.0, 8.5, 8.0, 9.0],
56           [45, 3.2, 8.0, 9.0, 9.0, 9.0, 8.0, 9.0, 8.5, 8.5, 9.5, 9.0, 8.5],
57           [83, 2.1, 8.5, 8.5, 9.0, 9.5, 8.0, 8.5, 9.0, 9.0, 8.5, 8.5, 9.0]
58          ]
59
60 round4 = [[17, 3.1, 8.5, 9.5, 9.5, 9.5, 9.0, 9.0, 8.5, 8.5, 8.0, 9.5, 9.0],
61           [23, 2.2, 9.0, 8.5, 8.5, 9.5, 8.5, 9.5, 9.0, 9.0, 8.0, 9.0, 8.5],
62           [28, 1.7, 8.0, 9.0, 9.5, 8.5, 8.0, 9.0, 8.0, 8.5, 9.0, 9.5, 8.5],
63           [45, 1.9, 8.0, 9.5, 8.0, 8.0, 8.5, 8.5, 9.5, 8.0, 9.5, 8.5, 9.0],
64           [83, 2.7, 8.5, 8.5, 9.0, 9.0, 9.0, 9.0, 9.0, 8.5, 9.5, 9.0, 8.5]
65          ]
66
67
68 # -----
69 # Subprograms
70 # -----
71 # =====> Write your code here
72 def show_leader_board(p_leaders):
73     out_board = []
74
75     # Initialise an empty displayable score board with all blanks
76     for count in range(len(p_leaders)):
77         out_board_row = list(NO_SCORE)           # For team number
78         out_board_row.append(NO_SCORE)           # For team name
79         out_board_row.append(NO_SCORE)           # For the total
80
81         # One column for each of the dives in the competition
82         for columns in range(NUM_ROUNDS):
83             out_board_row.append(NO_SCORE)
84         out_board.append(out_board_row)
85
86     # Figure out how many dives have scores (between 1 and 5)
87     # First field is not a dive, it's a team number
88     num_dives = len(p_leaders[0]) - 1
89
90     # Note - the contents of the out_board are not yet sorted in order
91     # Pick up information from the input parameter list holding
92     # the actual scores and put them into the empty score board
93     # Add up the total of existing dives per team for last column
94     out_board_row = 0           # Start at the top

```

```

95     for team in p_leaders:
96         total = 0                                # Reset each team
97         out_board[out_board_row][0] = team[0]    # Team number
98
99         # Look up team name in dictionary
100        out_board[out_board_row][1] = teams.get(team[0])
101
102        for count in range(1, num_dives + 1):
103            out_board[out_board_row][count + 1] = team[count]    # Dive score
104            total = total + team[count]    # Run team total
105
106        # Put total in last column
107        out_board[out_board_row][-1] = total
108
109        # Next slot on board
110        out_board_row = out_board_row + 1
111
112        # Sort on last column in reverse, to get winner at top
113        # Can use <list>.sort() because in place is fine on this copy
114        index_total = len(out_board[0]) - 1
115        out_board.sort(key = itemgetter(index_total), reverse = True)
116
117        # Display in score board headers
118        print("\n")
119        print(f"{titles[0]:^6} {titles[1]:^10} {titles[2]:^8} {titles[3]:^8} {titles[4]:^8} {titles[5]:^8} {titles[6]:^8}")
120
121        print("-"*62)
122        # Display the information for each row in the sorted score board
123        # Build up the formatting string for team number and team name
124        # Only for the number of dives with scores, pick up the score
125        # For remaining dives put in the NO_SCORE symbol
126        # Finish with the total, which all teams will have
127        for team in out_board:
128            layout_row = f"{team[0]:^6} {team[1]:^10} "    # Integer team
129                number and team name
130            for count in range(num_dives):    # Floating point
131                scores
132                layout_row = layout_row + f"{team[2 + count]:^8.2f} "
133            for count in range(NUM_ROUNDS - num_dives):    # Hyphen
134                layout_row = layout_row + f"{NO_SCORE:^8} "
135            layout_row = layout_row + f"{team[6]:^8.2f}"    # Total
136            print(layout_row)

```

```

136 def update_scores(p_team, p_score, p_leaders):
137     found = False
138     index = 0
139
140     # Find the team in the leader board
141     # Add the new score to the end
142     while ((index < len(p_leaders)) and (not found)):
143         if (p_leaders[index][0] == p_team):
144             p_leaders[index].append(p_score)
145             found = True
146         else:
147             index = index + 1
148
149 def calc_dive_score(p_dive_data, p_leaders):
150     count = 0
151
152     while (count < len(p_dive_data)):          # For each of five teams
153
154         # Work with one dive at a time
155         this_record = p_dive_data[count]
156
157         # Calculate raw score from all 11 judges
158         # Sort each set of judges scores, then pick up the middle
159         # one for diver1 and diver2, and the middle three for synchro
160         median_diver1 = (sorted(this_record[DIVER_1:DIVER_1 + 3]))[1]
161         median_diver2 = (sorted(this_record[DIVER_2:DIVER_2 + 3]))[1]
162         synchro_score = (sorted(this_record[SYNCHRO:SYNCHRO + 5]))[1:4]
163         raw_score = median_diver1 + median_diver2 + sum(synchro_score)
164
165         # Multiply by a scale adjustment
166         raw_score = raw_score * SCALE_ADJ
167
168         # Adjust for the degree of difficulty
169         round_score = raw_score * this_record[DIFFICULTY]
170
171         # Round to two places
172         total_score = round(round_score, 2)
173
174         # Update leader board with team id and new score
175         update_scores(this_record[TEAM], total_score, p_leaders)
176
177         count = count + 1
178
179 # -----
180 # Main program
181 # -----
182 # =====> Write your code here
183 calc_dive_score(round1, leader_board)
184 show_leader_board(leader_board)
185
186 calc_dive_score(round2, leader_board)
187 show_leader_board(leader_board)
188
189 calc_dive_score(round3, leader_board)
190 show_leader_board(leader_board)
191
192 calc_dive_score(round4, leader_board)
193 show_leader_board(leader_board)
194
195

```

Pearson Edexcel International Advanced Level

Sample assessment material

Paper
reference

WCP04/01

Computer Science

**Unit 4: Advanced Practical Programming and
Problem-solving
Programming Language Subset (PLS)**

Resource Booklet

Do not return this Booklet with the question paper.

S89029A

©2025 Pearson Education Ltd.
1/1/1/1/1/1/1



Introduction

The Programming Language Subset (PLS) is a document that specifies which parts of Python 3 are required in order that the assessments can be undertaken with confidence. Learners familiar with everything in this document will be able to access all parts of the Unit 4 assessment. This does not stop a teacher/learner from going beyond the scope of the PLS into techniques and approaches that they may consider to be more efficient or engaging.

Pearson will not go beyond the scope of the PLS when setting assessment tasks. Any learner successfully using more esoteric or complex constructs or approaches not included in this document will still be awarded marks in Unit 4, if the solution is valid.

The pair of <> symbols indicates where expressions or values need to be supplied. They are not part of the PLS.

Programming

Flow control constructs

Sequence

Every instruction comes one after the other, from the top of the file to the bottom of the file.

Selection

<pre>if <expression>: <command></pre>	If <expression> is true, then <command> is executed.
<pre>if <expression>: <command> else: <command></pre>	If <expression> is true, then first <command> is executed, otherwise second <command> is executed.
<pre>if <expression>: <command> elif <expression>: <command> else: <command></pre>	If <expression> is true, then first <command> is executed, otherwise the second <expression> test is checked. If true, then second <command> is executed, otherwise third <command> is executed. Supports multiple instances of elif. The else is optional.

Match-case statement

<pre>match <variable>: case <pattern 1>: <command> case <pattern 2>: <command> . . . case other: # Default <command></pre>	If <variable> matches <pattern 1>, then first <command> is executed. If <variable> matches <pattern 2>, then the second <command> is executed. Supports multiple instances of case <pattern>: The last case is the default, when no previous pattern has been matched. The last case is optional. It can be written as case _:
--	---

Count-controlled iteration

for <id> in <structure>: <command>	Executes <command> for each element of <structure>, in one dimension.
for <id> in range (<start>, <stop>): <command>	Executes <command> a fixed number of times, based on the numbers generated by the range function. <stop> is required. <start> is optional.
for <id> in range (<start>, <stop>, <step>): <command>	Same as above, except that <step> influences the numbers generated by the range function. <stop> is required. <start> and <step> are optional.

Condition-controlled iteration

while <condition>: <command>	Pre-conditioned loop. This executes <command> as long as <condition> is true.
---------------------------------	---

Exception handling

try: <command> except: <command>	The <command> under the try, is executed. If an error occurs, the <command> under the except is executed. Where there is no error, the code following the try/except block is executed.
try: <command> except <error type>: <command> except: <command>	The <command> under the try, is executed. If an error of <error type> occurs, the <command> under the except <error type> is executed. Otherwise, the code under the except is executed. Where there is no error, the code following the try/except block is executed.

Exception	Description
ArithmeticError	An error occurs in numeric calculations
AttributeError	An attribute reference or assignment fails
EOFError	The input () method hits an end-of-file condition (EOF)
Exception	The base exception class to catch a wide range of errors, rather than a specific error.
FileNotFoundError	The requested file does not exist
IndexError	An index of a sequence does not exist
KeyError	A key does not exist
NameError	A variable does not exist
TypeError	An attempt to combine two different types
UnboundLocalError	A local variable is referenced before assignment
ValueError	A wrong value in a specified data type
ZeroDivisionError	The second operator in a division is zero

Variables and constants

Identifiers are any sequence of letters, digits and underscores, starting with a letter.

Both uppercase and lowercase are supported.

Constants are conventionally named in all uppercase characters.

Assignment is used to set or change the value of an identifier.

<code><identifier> = <value></code>	Sets the <code><identifier></code> to the <code><value></code>
<code><identifier> = <expression></code>	Sets the <code><identifier></code> to the result of <code><expression></code>

Operators

Arithmetic

Arithmetic operator	Description
/	Division
*	Multiplication
**	Exponentiation
+	Addition
-	Subtraction
//	Integer division
%	Modulus

Relational

Relational operator	Description
==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Boolean

Boolean operator	Description
not	Inverts the argument.
and	The test on both sides of the operator must be true to return true.
or	The test on one side of the operator must be true to return true.

Bitwise

Bitwise operator	Name	Description
&	AND	For each pair of bits in two arguments, it returns a one only when both bits are one.
	OR	For each pair of bits in two arguments, it returns a one where either one or both bits are one.
^	XOR	For each pair of bits in two arguments, it returns a one where either one but not both bits are one.
~	NOT	Inverts
<<	Logical shift left	Moves the bits left by the given number of positions, filling with zeros from the right.
>>	Arithmetic shift right	Moves the bits right by the given number of positions, filling with the most-significant bit on the left.

Subprograms

def <procname> (): <command>	A procedure with no parameters.
def <procname> (<paramA>, <paramB>): <command>	A procedure with parameters.
def <funcname> (): <command> return (<value>)	A function with no parameters.
def <funcname> (<paramA>, <paramB>): <command> return (<value>)	A function with parameters.

Organising and handling data

Data types

Identifiers may be explicitly assigned a data type during declaration.

Identifiers may be implicitly assigned a data type during initialisation.

Generic data type	Implemented data type	Conversion
Integer	int	int()
Real	float	float()
Character	str	str()
Boolean	bool	bool()
String	str	str()

Type conversion

bool (<item>)	Returns <item> converted to the equivalent Boolean value.
float (<item>)	Returns <item> converted to the equivalent real value.
int (<item>)	Returns <item> converted to the equivalent integer value.
str (<item>)	Returns <item> converted to the equivalent string value.
type (<obj>)	Returns the type of <obj>.

Data structures

A data structure is a sequence of items, which themselves are typed.

Where a data structure supports indexing, indexes start with zero.

Structure	Description	Python data type
Array	A sequence of items with the same (homogeneous) data type. Ordered. Mutable. Duplicates allowed.	list
Dictionary	Data is stored in key:value pairs. Ordered. Mutable. Duplicate keys not allowed.	dict
List	A sequence of items with the same (homogeneous) or mixed (heterogenous) data types. Ordered. Mutable. Duplicates allowed.	list
Record	A sequence of items usually of mixed (heterogenous) data types.	list
Set	A sequence of items with the same (homogeneous) or mixed (heterogenous) data types. Unordered. Mutable. Duplicates not allowed.	set
String	A sequence of characters. Ordered. Immutable. Duplicates allowed.	str
Tuple	A sequence of items with the same (homogeneous) or mixed (heterogenous) data types. Ordered. Immutable. Duplicates allowed.	tuple

Dimensions

<struct>[<dim_one>]	Accesses an item in a one-dimensional structure.
<struct>[<dim_one>][<dim_two>]	Accesses an item in a two-dimensional structure.

Ragged data structures are not supported. therefore, in a list of records, each record will have the same number of fields.

Abstract data types

An abstract data type is a complex data type, implemented using a simpler data structure. For example, a stack is an abstract data type. It can be implemented as a list.

List

Lists can be:

- indexed using []
- concatenated using the + operator
- sliced using the : operator.

<code><aList> = list ()</code> <code><aList> = []</code>	Two methods of creating a list structure. Both are empty.
<code><list>.append (<item>)</code>	Adds <item> to the end of <list>.
<code><list>.clear ()</code>	Removes all items from <list>.
<code><list>.copy ()</code>	Returns a new object with the same contents as <list>.
<code><list>.count (<value>)</code>	Returns the number of items with <value>.
<code>del <list> [<index>]</code>	Removes the item at <index> from <list>.
<code><list>.index (<value>)</code>	Returns the index of the first <value>.
<code><list>.insert (<index>, <item>)</code>	Inserts <item> just before an existing one at <index>.
<code><list>.pop (<index>)</code>	Removes and returns the item at <index> from <list>. <index> is optional. The default is -1, which returns the last item appended to <list>.
<code><list>.reverse ()</code>	Reverses the contents of <list>.

<list>.sort (<option>)	Sorts <list> in place.	
	Option	Effect
	blank	All defaults, i.e. ascending and case sensitive.
	reverse = True	Sorts in descending order.
	key = <afunction>	Sorts <object> based on the returned value from <afunction>, which may be a lambda function.
	key = str.lower()	Makes the sort case insensitive.
<list>.remove (<value>)	Removes the first item with a value of <value>.	

Dictionary

<aDict> = dict (<key1> = <value1>, <key2> = <value2>)	Creates a dictionary with two key:value pairs, using the dict () constructor.
<aDict> = {<key1>: <value1>, <key2>: <value2>}	Creates a dictionary with two key:value pairs.
<aDict>[<new key>] = <new value>	Causes a <new key>:<new value> pair to be added to <aDict>.
<aDict>[<key>] = <new value>	Changes the value associated with <key> in <aDict> to <new value>.
<aDict>.clear ()	Empties <aDict>.
del <aDict>[<key>]	Removes the <key> and its associated value from <aDict>.
<aDict>.get (<key>)	Returns the value associated with <key> in <aDict>.
<aDict>.items ()	Returns an object of tuples. Each tuple is a key:value pair. Use list () to convert the returned result to a list.
<aDict>.keys ()	Returns all the keys in <aDict>. Use list () to convert the returned result to a list.
<aDict>.values ()	Returns all the values in <aDict>. Use list () to convert the returned result to a list.

Tuple

Tuples can be:

- indexed using []
- concatenated using the + operator
- sliced using the : operator.

<code><aTuple> = (<item>,) </code>	Creates a tuple of one item. An extra comma is needed. Otherwise, the single <item> will be treated as a different data type.
<code><aTuple> = tuple ()</code>	Creates an empty tuple.

Set

<code><aSet> = {<value1>, <value2>}</code>	Creates a set with two items. The items can be different data types.
<code><aSet> = set ((<value1>, <value2>))</code>	Creates a set with the set () constructor. Two sets of round brackets are required. The items can be different data types.
<code><aSet>.add (<value>)</code>	Adds <value> to the existing set <aSet>.
<code><aSet>.clear ()</code>	Empties <aSet>.
<code><set1>.difference (<set2>)</code>	Returns the set of items that exist only in <set1> but not in <set2>. This is the difference operation for sets.
<code><aSet>.discard (<value>)</code>	Removes <value> from <aSet>. If <value> is not in the set, an error will not occur.
<code><value> in <aSet></code>	Returns true, if the <value> is in <aSet>. Otherwise, returns false. This is the membership operation for sets.
<code>set.intersection (<set1>, <set2>)</code>	Returns a new set with all the items that occur in both sets. This is the intersection operation for sets.
<code><aSet>.remove (<value>)</code>	Removes <value> from <aSet>. If <value> is not in the set, an error will occur.
<code><set1>.symmetric_difference (<set2>)</code>	Returns a new set with all items that are in <set1> or in <set2>, but not in both sets. This is the exclusive or operation for two sets.
<code><set1>.union (set2)</code>	Returns a new set with all the items from both sets. This is the union operation for sets.

Set operator	Description
-	Difference
&	Intersection
^	Symmetric difference (Exclusive or)
	Union

Numeric data

<code>chr (<integer>)</code>	Returns the string which matches the Unicode value of <integer>. The first 128 characters of Unicode are equivalent to ASCII.
<code>int (<item>)</code>	Returns <item> converted to the equivalent integer value. This removes the fractional part of <item>.
<code>range (<start>, <stop>, <step>)</code>	Generates a list of numbers using <step>, beginning with <start> and up to, but not including, <stop>. A negative value for <step> goes backwards. <stop> is required. <start> and <step> are optional. The default value for <start> is zero.
<code>round (<x>, <n>)</code>	Rounds <x> to the number of <n> digits after the decimal (uses the 0.5 rule). The <n> is optional. If omitted, the function returns the nearest integer to <x>.

Decimal library

The decimal library module is used for high-precision floating-point numbers. Using the decimal module prevents the precision loss that can happen when using floating-point numbers.

<code>decimal.Decimal (<value>)</code>	Creates a decimal object with the value of <value>.
<code>decimal.getcontext()</code>	Returns the current context for the decimal library module.
<code>decimal.getcontext ().prec = <value></code>	Sets the output precision <value> for the decimal module.
<code><object>.sqrt ()</code>	Returns the square root of the decimal <object>.
<code>decimal.compare (<x>, <y>)</code>	Returns -1, when <x> is less than <y>. Returns 1, when <x> is greater than <y>. Returns 0, when <x> is equal to <y>.
<code><x>.copy_abs ()</code>	Returns the absolute value of the decimal number <x>.
<code><x>.min (<y>)</code>	Returns the minimum of <x> and <y>.
<code><x>.max (<y>)</code>	Returns the maximum of <x> and <y>.
<code><x>.logical_and (<y>)</code>	Returns the result of a logical AND on the two decimal numbers <x> and <y>.
<code><x>.logical_or (<y>)</code>	Returns the result of a logical OR on the two decimal numbers <x> and <y>.
<code><x>.logical_xor (<y>)</code>	Returns the result of a logical XOR on the two decimal numbers <x> and <y>.

String data

Strings can be:

- indexed using []
- concatenated using the + operator
- sliced using the : operator
- repeated using the * operator.

<code><string>.capitalize ()</code>	Returns a new string, equivalent to <code><string></code> where the first letter is capitalised.
<code><string>.find (<substring>, <start>, <end>)</code>	Returns the location of the first instance of <code><substring></code> in the original <code><string></code> , reading from left to right. <code><start></code> is the index to begin the find. The default is zero. <code><end></code> is the index to stop the find. The default is the end. <code><string></code> . Returns -1, if not found.
<code><string>.format (<values>)</code>	Formats <code><values></code> and puts them into <code><string></code> . The content of <code><string></code> is described by symbols and placeholders.
<code><string>.index (<substring>, <start>, <end>)</code>	Returns the location of the first instance of <code><substring></code> found in the original <code><string></code> as read from left to right. Raises an exception if not found. <code><substring></code> is required. <code><start></code> and <code><end></code> are optional. The default value for <code><start></code> is zero. The default value for <code><end></code> is the end of <code><string></code> .
<code><string>.isalnum ()</code>	Returns true, if all characters are alphabetic A–Z, a–z or digits 0–9.
<code><string>.isalpha ()</code>	Returns true, if all characters are alphabetic A–Z or a–z.
<code><string>.isdigit ()</code>	Returns true, if all characters are digits 0–9, exponents are digits.
<code><string>.islower ()</code>	Returns true, if all characters in <code><string></code> are lowercase.
<code><string>.isnumeric ()</code>	Returns true if all the characters in <code><string></code> are the values 0 to 9.
<code><string>.isspace ()</code>	Returns true if all the characters in <code><string></code> are whitespaces.
<code><string>.isupper ()</code>	Returns true, if all characters in <code><string></code> are uppercase.
<code>len (<string>)</code>	Returns the length of <code><string></code> .
<code><string>.lower ()</code>	Returns original <code><string></code> in lowercase.

<code>ord (<char>)</code>	Returns the integer equivalent to the Unicode string of the single character <char>. The first 128 characters of Unicode are equivalent to ASCII.
<code><string>.strip (<char>)</code>	Returns original <string> with all occurrences of <char> removed from the front and back.
<code><string>.split (<char>)</code>	Returns a list of all substrings in original <string>, using <char> as the separator.
<code><string>.title ()</code>	Returns a string where the first character of each word is converted to upper case.
<code><string>.upper ()</code>	Returns original <string> in uppercase.

Formatted string literals (f-strings)

The value of expressions can be included inside a string, where the string has been prefixed with the character `f` or the character `F`. The expressions are supplied inside curly braces, `{ }`. An optional format specifier may follow each expression, separated from the expression by a semi-colon.

`f"<expr>:<align><sign><width><.precision><type>}"`

Placeholder	Option	Description
align	<	Left aligned. Default for most items, such as text.
	>	Right aligned. Default for numbers.
	^	Centre aligned.
sign	+	Use a sign for both positive and negative numbers.
	-	Use a sign only for negative numbers. Default for negative numbers.
	space	Use leading spaces for positive numbers and a minus sign for negative numbers.
width	whole number	The total width of the field.
precision	whole number	The number of digits after the decimal.
type	s	String. Default for strings, if not supplied.
	d	Numbers in base 10 (denary). Default for integers, if not supplied.
	f	Fixed-point notation. Formats a number with exactly the number of digits to the right of the decimal given by precision.

Date and time

Dates and times follow the ISO 8601 standard, designed to eliminate ambiguity and support worldwide exchange of data. The ordering, field lengths and separators are fixed.

String	ISO 8601 Format
2024-09-27	YYYY-MM-DD
14:30:00	HH:MM:SS
24-09-27 14:30:00	YYYY-MM-DD HH:MM:SS

All dates and times should be processed as Python datetime objects. No localisation of dates and times is required for processing or presentation.

The Python format string to describe the presentation of ISO 8601 format is: "%Y-%m-%d %H:%M:%S".

Storing and processing dates and times	
datetime library	Library that contains subprograms to manipulate a datetime object that represents Coordinated Universal Time (UTC).
Text files	Store as a text string in ISO 8601 format.
pandas library	Loading CSV files • Read CSV data into a dataframe, using <code>pandas.read_csv ()</code> • Convert date and time columns to datetime object, using <code>pandas.to_datetime (<df_column>)</code> Saving CSV files • Convert datetime object columns to string format, using <code><df_column>.dt.strftime ()</code> with an argument of <code>"%Y-%m-%d %H:%M:%S"</code> • Write the dataframe to CSV using <code><dataframe>.to_csv ()</code>

RegEx

<code>re.findall (<target>, <text>)</code>	Returns a list holding all found matches for <target> in <text>.						
<code>re.search (<target>, <text>)</code>	Returns a Match object if there is a match anywhere in the string. <table border="1"> <thead> <tr> <th>Match attribute</th><th>Description</th></tr> </thead> <tbody> <tr> <td><code>.start</code></td><td>Index in <text> for the first letter of <target>.</td></tr> <tr> <td><code>.end</code></td><td>Index in <text> one past the last letter of <target>.</td></tr> </tbody> </table>	Match attribute	Description	<code>.start</code>	Index in <text> for the first letter of <target>.	<code>.end</code>	Index in <text> one past the last letter of <target>.
Match attribute	Description						
<code>.start</code>	Index in <text> for the first letter of <target>.						
<code>.end</code>	Index in <text> one past the last letter of <target>.						
<code>re.split (<chars>, <text>)</code>	Returns a list where <string> is split at each match of <chars>.						
<code>re.sub (<target>, <chars>, <text>)</code>	Replaces one or many matches of <target> in <text> with <chars>.						

Metacharacters

Character	Description
[]	Indicates a set of characters.
\	Indicates a special sequence. Also used as for escape characters.
.	Matches any character, except a newline character.
^	Matches at the beginning of the line, as in starts with. Used as a NOT operator.
\$	Matches at the end of the line, as ends with.
*	Zero or more repetitions of the preceding regular expression.
+	One or more repetitions of the preceding regular expression.
?	Zero or one repetitions of the preceding regular expression.
{<options>}	{<m>} – match exactly <m> copies of the preceding regular expression. {<m>, <n>} – match from <m> to <n> copies of the preceding regular expression.
	The OR operator, as in either one or the other.
()	Referred to as a capture group that allows extraction of specific parts of a match.

Character sets

Character set	Description
[<char>...<char>]	Returns a match for any individual <char> in the list.
[<start>-<end>]	Returns a match for any character between <start> and <end>.
[^<char>...<char>]	Returns a match for any individual <char> not in the list.
[<s1>-<s2>] [<s3>-<s4>]	Returns a match for any two-character string where the first character is from the range defined by <s1> and <s2> and the second character is from the range defined by <s3> and <s4>.
[a-zA-Z]	Returns a match for any character from a to z or from A to Z.
[+]	Returns a match for any + symbol in the text.

Special sequences

Character	Description
\A	Matches only if the specified characters are at the beginning of the text.
\b	Matches where the characters are at the beginning or end of the text. Precede the target with r, as in r"<target>".
\B	Matches where the target is found in the text, but NOT at the beginning or at the end of the text. Precede the target with r, as in r"<target>".
\d	Matches any decimal digit, [0-9].
\D	Matches any non-digit character, ^[0-9].
\s	Matches any whitespace character, [\t\n\r\f\v].
\S	Matches any non-whitespace character, [^\t\n\r\f\v].
\w	Matches any alphanumeric character, [a-zA-z0-9_].
\W	Matches any non-alphanumeric character, [^a-zA-z0-9_].
\Z	Matches if the target is at the end of the text.

Text files

Where lines in the file contain more than a single column of data, a comma is used as a delimiter.

<code><fileid> = open (<filename>, "r")</code>	Opens file for reading.
<code>for <line> in <fileid>:</code>	Reads every line, one at a time.
<code><alist> = <fileid>.readlines ()</code>	Returns a list where each item is a line from the file.
<code><aline> = <fileid>.readline ()</code>	Returns a line from a file. Returns an empty string on the end of the file.
<code><fileid> = open (<filename>, "w")</code>	Opens a file for writing.
<code><fileid> = open (<filename>, "a")</code>	Opens a file for appending.
<code><fileid>.writelines (<struct>)</code>	Writes <structure> to a file. <struct> is a list of strings.
<code><fileid>.write (<aString>)</code>	Writes a single string to a file.
<code><fileid>.close ()</code>	Closes the file.

Control characters

Name	Abbreviation	ASCII hexadecimal	String
Carriage return	CR	0x0D	"\r"
Line feed	LF	0x0A	"\n"
Horizontal tab	HT	0x09	"\t"

Best practice

Global keyword

The PLS supports considered and minimal use of the global keyword.

Blocked code

Blocking of code segments is indicated by indentation and subprogram calls. These determine the scope and extent of variables they declare.

Built-in subprograms

<code>input (<prompt>)</code>	Displays <prompt> on the screen and returns the line typed in.										
<code>len (<object>)</code>	Returns the length of the <object>										
<code>print (<item>)</code>	Displays <item> on the screen.										
<code>sorted (<object>, <option>)</code>	Returns a sorted list of items in <object>. The original <object> is not changed. <table><tr><th>Option</th><th>Effect</th></tr><tr><td><code>blank</code></td><td>All defaults, i.e. ascending and case sensitive.</td></tr><tr><td><code>reverse = True</code></td><td>Sorts in descending order.</td></tr><tr><td><code>key = <afunction></code></td><td>Sorts <object> based on the returned value from <afunction>, which may be a lambda function.</td></tr><tr><td><code>key = str.lower()</code></td><td>Makes the sort case insensitive.</td></tr></table>	Option	Effect	<code>blank</code>	All defaults, i.e. ascending and case sensitive.	<code>reverse = True</code>	Sorts in descending order.	<code>key = <afunction></code>	Sorts <object> based on the returned value from <afunction>, which may be a lambda function.	<code>key = str.lower()</code>	Makes the sort case insensitive.
Option	Effect										
<code>blank</code>	All defaults, i.e. ascending and case sensitive.										
<code>reverse = True</code>	Sorts in descending order.										
<code>key = <afunction></code>	Sorts <object> based on the returned value from <afunction>, which may be a lambda function.										
<code>key = str.lower()</code>	Makes the sort case insensitive.										

Libraries of subprograms

The functionality of a library module can only be accessed once the library module is imported into the program code.

<code>import <library></code>	Imports the whole of the <library> module into the current program.
<code>import <library> as <alias></code>	Imports the whole of the <library> which can be addressed as <alias>.
<code>from <library> import <identifier></code>	Imports only <identifier> from the <library> module.
<code>from <library> import <identifier> as <alias></code>	Imports only <identifier> from the <library> module which can be addressed <alias>.

Random

<code>random.randint (<a>,)</code>	Returns a random integer X so that $\langle a \rangle \leq X \leq \langle b \rangle$.
<code>random.random ()</code>	Returns a floating-point number in the range of 0.0 to 1.0
<code>random.shuffle (<list>)</code>	Randomises the contents of <list>, in place.
<code>random.uniform (<a>,)</code>	Returns a random floating-point number X so that $\langle a \rangle \leq X < \langle b \rangle$

Math

<code>math.ceil (<x>)</code>	Returns the smallest integer greater than or equal to <x>.
<code>math.fabs (<x>)</code>	Returns the absolute value of <x>. Equivalent to $ \langle x \rangle $.
<code>math.floor (<x>)</code>	Returns the largest integer less than or equal to <x>.
<code>math.pi</code>	The constant Pi (Π).
<code>math.pow (<x>, <y>)</code>	Returns the value of <x> raised to the power of <y>.
<code>math.sqrt (<x>)</code>	Returns the square root of <x>.
<code>math.trunc (<x>)</code>	Returns the integer part of <x>.

Copy

<code>copy.copy (<obj>)</code>	Returns a shallow copy of <obj>.
<code>copy.deepcopy (<obj>)</code>	Returns a deep copy of <obj>.

NumPy

<code><arr>[<dim1>][<dim2>] ...</code>	Accesses the item in <arr> found at index values given by <dim1> and <dim2>. Index may be negative.
<code><arr>[<start>:<stop>:<step>]</code>	Array slicing. <start> is the beginning, up to but not including <stop>, by <step> interval.
<code><arr>[<filter>]</code>	Array filtering. Selects only those items in <arr> where the corresponding value in <filter> is true.

<code>numpy.array (<obj>, <type>)</code>	Returns a numpy ndarray made from items in <obj>. <type> is optional and may have these values: <table><tr><th><type></th><th>Type</th></tr><tr><td>'i'</td><td>int</td></tr><tr><td>'b'</td><td>bool</td></tr><tr><td>'f'</td><td>float</td></tr><tr><td>'M'</td><td>Datetime</td></tr><tr><td>'S'</td><td>str</td></tr></table>	<type>	Type	'i'	int	'b'	bool	'f'	float	'M'	Datetime	'S'	str
<type>	Type												
'i'	int												
'b'	bool												
'f'	float												
'M'	Datetime												
'S'	str												
<code>numpy.array ([<filter>])</code>	Returns a ndarray containing only the items where the corresponding item in <filter> is true or the expression <filter> evaluates to true.												

<code><arr1> <arith> <arr2></code>	Returns a ndarray with items indicating the evaluation of <code><arr1> <arith> <arr2></code>. <table border="1" data-bbox="699 241 1326 622"> <thead> <tr> <th data-bbox="699 241 1007 315"><arith></th><th data-bbox="1011 241 1326 315">Value</th></tr> </thead> <tbody> <tr> <td data-bbox="699 322 1007 396">+</td><td data-bbox="1011 322 1326 396">Addition</td></tr> <tr> <td data-bbox="699 403 1007 477">-</td><td data-bbox="1011 403 1326 477">Subtraction</td></tr> <tr> <td data-bbox="699 483 1007 557">*</td><td data-bbox="1011 483 1326 557">Multiplication</td></tr> <tr> <td data-bbox="699 564 1007 622">/</td><td data-bbox="1011 564 1326 622">Division</td></tr> </tbody> </table>	<arith>	Value	+	Addition	-	Subtraction	*	Multiplication	/	Division
<arith>	Value										
+	Addition										
-	Subtraction										
*	Multiplication										
/	Division										
<code><arr>.astype (<type>)</code>	Returns a new array based on the items in <code><arr></code> with the data types converted to <code><type></code>.										
<code>numpy.average (<arr1>, weights = <values>)</code>	Returns the arithmetic mean of all items in <code><arr1></code>, when weights is omitted. Where <code><values></code> is an ndarray of weights, the weight is applied to each item <code><arr1></code> during the calculation.										
<code>numpy.column_stack (<arr1>, ...)</code>	Stacks a sequence of one or more one-dimensional arrays as columns into a two-dimensional array. All of the arrays must have the same first dimension.										
<code>numpy.loadtxt (<file>, <arg1>, ...)</code>	Returns a ndarray constructed from the data in <code><file></code>. Additional arguments are optional and may have these values: <table border="1" data-bbox="699 1285 1326 1711"> <thead> <tr> <th data-bbox="699 1285 895 1359"><arg></th><th data-bbox="900 1285 1326 1359">Description</th></tr> </thead> <tbody> <tr> <td data-bbox="699 1366 895 1494">delimiter</td><td data-bbox="900 1366 1326 1494">The character used to separate the values in the file. The default is a space.</td></tr> <tr> <td data-bbox="699 1500 895 1574">dtype</td><td data-bbox="900 1500 1326 1574">The data type of the resulting ndarray. The default is float.</td></tr> <tr> <td data-bbox="699 1581 895 1711">skiprows</td><td data-bbox="900 1581 1326 1711">Skips the number of rows specified from the beginning of the file. The default is 0.</td></tr> </tbody> </table>	<arg>	Description	delimiter	The character used to separate the values in the file. The default is a space.	dtype	The data type of the resulting ndarray. The default is float.	skiprows	Skips the number of rows specified from the beginning of the file. The default is 0.		
<arg>	Description										
delimiter	The character used to separate the values in the file. The default is a space.										
dtype	The data type of the resulting ndarray. The default is float.										
skiprows	Skips the number of rows specified from the beginning of the file. The default is 0.										
<code>numpy.mean (<arr1>, axis = <num>)</code>	Returns the arithmetic mean of all items in <code><arr1></code>, along axis number <code><num></code>. Axis is optional. The default for axis is 0.										
<code>numpy.mod (<arr1>, <arr2>)</code> <code>numpy.remainder (<arr1>, <arr2>)</code>	Returns a ndarray with the remainders of all <code><arr1></code> divided by corresponding item in <code><arr2></code>.										

<code>numpy.ndmin (<arr>)</code>	Returns the rank (number of dimensions) in <arr>.														
<code>numpy.power (<arr1>, <arr2>)</code>	Returns a ndarray with each item in <arr1> raised to the power of the corresponding item in <arr2>.														
<code>numpy.ravel (<arr>)</code>	Returns a flattened ndarray of all the items in <arr>. *<arr> is a shortcut for ravel.														
<code><filter> = <arr1></code> <code> <relation></code> <code> <arr2></code>	Returns a Boolean ndarray with items indicating the evaluation of <arr1> <relation> <arr2>. <table border="1"> <thead> <tr> <th><relation></th><th>Description</th></tr> </thead> <tbody> <tr> <td>></td><td>Greater than</td></tr> <tr> <td>>=</td><td>Greater than or equal to</td></tr> <tr> <td>==</td><td>Equal to</td></tr> <tr> <td>!=</td><td>Not equal to</td></tr> <tr> <td><=</td><td>Less than or equal to</td></tr> <tr> <td><</td><td>Less than</td></tr> </tbody> </table>	<relation>	Description	>	Greater than	>=	Greater than or equal to	==	Equal to	!=	Not equal to	<=	Less than or equal to	<	Less than
<relation>	Description														
>	Greater than														
>=	Greater than or equal to														
==	Equal to														
!=	Not equal to														
<=	Less than or equal to														
<	Less than														
<code>numpy.reshape (<arr>, <shape>)</code>	Returns the contents of <arr> reshaped to the dimensions in <shape>. <arr> is an integer or tuple of integers. The original <arr> and the new <shape> must be compatible. Where <shape> is an integer, that number of elements of <arr> is returned. Where <shape> is -1, the result is all the elements of <arr>.														

```
numpy.savetxt (<file>,
               <arr>,
               <arg1>, ...)
```

Saves <arr> to <file>. Additional arguments are optional and may have these values:

<arg>	Description								
delimiter	The character used to separate the values in the file. The default is a space.								
fmt	A single string or multiple strings in a list that specify a format for the data in <arr>. Common strings are: <table border="1"> <thead> <tr> <th>String</th><th>Type</th></tr> </thead> <tbody> <tr> <td>"%d"</td><td>Integer</td></tr> <tr> <td>"%.2f"</td><td>Float to two decimal places</td></tr> <tr> <td>"%s"</td><td>String</td></tr> </tbody> </table>	String	Type	"%d"	Integer	"%.2f"	Float to two decimal places	"%s"	String
String	Type								
"%d"	Integer								
"%.2f"	Float to two decimal places								
"%s"	String								
header	A string that will be written at the top of <file>.								

```
numpy.shape (<arr>)
```

Returns a tuple with the same number of items as dimensions in <arr>, with each value indicating the number of items in that dimension.

```
numpy.size (<arr>, <axis>)
```

Returns the number of elements along the given <axis>. Where <axis> is omitted, the total number of elements is returned.

```
<arr>.sort ()
```

Sorts the items in <arr> in place. This destroys the original <arr>.

```
numpy.sort (<arr>)
```

Returns an ndarray of the sorted values in <arr>. This does not change the original <arr>.

<code>numpy.sort (<arr>, axis = <int>)</code>	Returns an ndarray of <arr>, sorted on the indicated axis. <table><tr><th><int></th><th>Description</th></tr><tr><td>0</td><td>Along the first axis.</td></tr><tr><td>1</td><td>Along the second axis.</td></tr><tr><td>-1</td><td>Along the last axis.</td></tr><tr><td>None</td><td>Along no axis, which results in a flattened <arr>.</td></tr></table>	<int>	Description	0	Along the first axis.	1	Along the second axis.	-1	Along the last axis.	None	Along no axis, which results in a flattened <arr>.
<int>	Description										
0	Along the first axis.										
1	Along the second axis.										
-1	Along the last axis.										
None	Along no axis, which results in a flattened <arr>.										
<code>numpy.std (<arr1>)</code>	Returns a measure of standard deviation. This is the average distance of the values in <arr1> from the mean of <arr1>, expressed in the same units as the original values.										
<code>numpy.sum (<arr1>, axis = <num>)</code>	Returns a single value for the sum of all items in <arr1>, along axis number <num>. Axis is optional. The default for axis is 0.										
<code><arr>.tolist ()</code>	Converts <arr> to a list.										
<code>numpy.transpose (<arr1>)</code>	Returns an ndarray, with the same values as <arr1>, where the columns and rows are swapped.										
<code>numpy.var (<arr1>)</code>	Returns a measure of variance. This is how much the group of values in <arr1> differs from the mean of the group. This indicates the spread of the values.										
<code>numpy.vstack (<arr1>, ...)</code>	Stacks a sequence of one or more one-dimensional arrays as rows into a two-dimensional array. All of the arrays must have the same first dimension.										
<code>numpy.where (<arr>, <condition>)</code>	Returns a ndarray of indexes, in <arr>, where <condition> is true.										
<code>numpy.where (<arr>, <condition>, <true>, <other>)</code>	Returns a ndarray of <true> from <arr> here <condition> is true, otherwise returns <other>.										

Pandas

Dataframes can be thought of as a two-dimensional table with rows and columns.

Series can be thought of as a one-dimensional array representing a single column.

Indexes can be thought of as the row labels. Dataframes without explicit row labels are addressed by indexes, starting at zero.

Dataframes

<code>DataFrame (<obj>)</code>	Returns a pandas dataframe made from items in <obj>. Where <obj> is a dictionary, the keys become the column names.
<code>DataFrame (<obj>, columns = <names>, index = <labels>)</code>	Returns a pandas dataframe made from items in <obj>, with column headers given in <names> and row identifiers given in <labels>. Both columns and index are optional.

`pandas.read_csv (<file>, <arg1>, ...)`

Returns the contents of <file> into a dataframe. Additional arguments are optional and may have these values:

<arg>	Description
delimiter	The character used to separate the values in each line of the file. The default is a comma.
dtype	The data type of the resulting dataframe or individual columns in the dataframe. The specifier may be a dictionary, where key is the column name and value is the data type. <pre>{'col1': str, 'col2': float, 'col3': int}</pre> Where this argument is not supplied, Pandas infers the type from the values in each column.
header	Specifies which row in the file will be used as the header for the columns. The default is line 0, which means the column names are inferred from the first line of the file.
skiprows	Skips the number of lines specified from the beginning of the file. The default is none.
<pre>parse_dates = [<col_name>] dayFirst = True</pre> Use to read European date and times from file into the dataframe. <col_name> is the header for the column that contains the date time string.	

`<df>[<column>]`

Returns a series of the data in <df> in the column named <column>. See the following section for information about series.

`<df>[<column>] = <obj>`

Adds a column named <column> to <df> with the values in <obj>.

<df>[<condition>]	Returns a dataframe with rows where <condition> is true.														
	<table><tr><th>Relation</th><th>Description</th></tr><tr><td>></td><td>Greater than</td></tr><tr><td>>=</td><td>Greater than or equal to</td></tr><tr><td>==</td><td>Equal to</td></tr><tr><td>!=</td><td>Not equal to</td></tr><tr><td><=</td><td>Less than or equal to</td></tr><tr><td><</td><td>Less than</td></tr></table>	Relation	Description	>	Greater than	>=	Greater than or equal to	==	Equal to	!=	Not equal to	<=	Less than or equal to	<	Less than
	Relation	Description													
	>	Greater than													
	>=	Greater than or equal to													
	==	Equal to													
	!=	Not equal to													
	<=	Less than or equal to													
<	Less than														
<df>.describe ()	Returns statistical information about <df>.														
<df>.drop (<row>, inplace = True)	Causes the <row> to be removed from <df>. The inplace key word cause the original <df> to be changed.														
<df>.drop_duplicates (inplace = <bool>)	Removes all the duplicate rows from <df>. The inplace keyword, which may be omitted, if set to true, changes the original <df>.														
<df>.dropna ()	Returns a new dataframe where each row with a null value is left out.														
<df>.duplicated ()	Returns a series reporting true for every row that is a duplicate and false for every row that is not a duplicate.														
<df>.head (<value>)	Returns the first <value> rows in <df>. Where.<value> is omitted, the first five rows are returned. Column headers are also returned.														
<df>.iloc[<value>]	Returns the item at the location indicated by <value>. <value> is integer-based and can include: A single integer A list of integers A slice of integers, where both the start and stop are included														
<df>.info ()	Returns information about <df>.														

<code><df>.loc[<value>]</code>	Returns the item at the location indicated by <value>. <value> is label-based and can include: • A single label • A list of labels • A slice of integers, where the stop is not included												
<code><df>.loc[<index>, <column>]</code>	Accesses value at <index>, <column> with a new <value>. Can be used to retrieve, change, or check a value.												
<code><df>.sort_values (</code> <code>by = <column>,</code> <code>ascending = <bool>)</code>	Returns a dataframe sorted by <column>. Where <bool> is true, the order is ascending. This is the default when ascending is omitted. For descending order, use ascending = false.												
<code><df>.tail (<value>)</code>	Returns the last <value> rows in the dataframe. Where <value> is omitted, the first five rows are returned. Column headers are also returned.												
<code><df>.to_csv (<file>,</code> <code><arg1>, ...)</code>	Saves <df> to a csv file with the name <file>. Additional arguments are optional and may have these values: <table border="1"> <thead> <tr> <th><arg></th><th>Description</th></tr> </thead> <tbody> <tr> <td>sep</td><td>The character used to separate the values in the file. The default is a comma.</td></tr> <tr> <td>float_format</td><td>Format string for floating point numbers, such as "%.2f", for two decimal places.</td></tr> <tr> <td>index</td><td>Writes row indexes to the file. The default is true.</td></tr> <tr> <td>header</td><td>Writes column headers to the file. The default is true.</td></tr> <tr> <td>date_format</td><td>Format string for datetime objects, such as "%d/%m/%Y %H:%M:%S"</td></tr> </tbody> </table>	<arg>	Description	sep	The character used to separate the values in the file. The default is a comma.	float_format	Format string for floating point numbers, such as "%.2f", for two decimal places.	index	Writes row indexes to the file. The default is true.	header	Writes column headers to the file. The default is true.	date_format	Format string for datetime objects, such as "%d/%m/%Y %H:%M:%S"
<arg>	Description												
sep	The character used to separate the values in the file. The default is a comma.												
float_format	Format string for floating point numbers, such as "%.2f", for two decimal places.												
index	Writes row indexes to the file. The default is true.												
header	Writes column headers to the file. The default is true.												
date_format	Format string for datetime objects, such as "%d/%m/%Y %H:%M:%S"												
<code><df>.to_datetime (<data>)</code>	Converts all the data in <data> to a datetime format. <data> is a column specified by <df>[<column>].												
<code><df>.to_string ()</code>	Returns the contents of <df> converted to strings, suitable for printing. Where the <df> is large, only the first five and the last five rows are returned.												

Series

<code>pandas.Series (<obj>, index = <labels>)</code>	Returns a pandas Series from items in <obj>. Where the index keyword is provided, <labels> represent the unique identifier for each row. The length of <labels> must match the length of <obj>. Where the index keyword is omitted, each row is assigned an integer value, beginning at position 0.										
<code>Series (<obj>, index = <labels>)</code>	Returns a pandas Series from items in <obj>. Where the index keyword is provided, <labels> represent the unique identifier for each row. The length of <labels> must match the length of <obj>. Where the index keyword is omitted, each row is assigned an integer value, beginning at position 0.										
<code><series>.apply (<func>)</code>	Applies <func> to each value.										
<code><series>.count ()</code>	Returns the count of non-null values.										
<code><series>.describe ()</code>	Where <series> is a column of strings from a dataframe, this method returns count of values, number of unique values, the most frequent value, and the frequency of the most common value.										
<code><series>.dt.strftime (<arg>)</code>	Returns an ndarray of formatted strings using the specified date format. This is equivalent to a column in a dataframe. <arg> is a string format for datetime, e.g. "%Y-%m-%d %H:%M:%S".										
<code><series1> <arith> <series2></code>	Returns a series with items indicating the evaluation of <series1> <arith> <series2>. <table border="1"> <thead> <tr> <th><arith></th><th>Description</th></tr> </thead> <tbody> <tr> <td>+</td><td>Addition</td></tr> <tr> <td>-</td><td>Subtraction</td></tr> <tr> <td>*</td><td>Multiplication</td></tr> <tr> <td>/</td><td>Division</td></tr> </tbody> </table>	<arith>	Description	+	Addition	-	Subtraction	*	Multiplication	/	Division
<arith>	Description										
+	Addition										
-	Subtraction										
*	Multiplication										
/	Division										
<code><series>[<label>]</code>	Returns the item in <series> at the position named <label>. Series without <labels> are accessed with integers, beginning at position 0.										
<code><series>.max ()</code>	Returns the largest value.										
<code><series>.mean()</code>	Returns the mean for all the values in column named <column>.										

<code><series>.median()</code>	Returns the median for all the values in column named <code><column></code> . Median is the middle item when the values in <code><column></code> are sorted into ascending order.
<code><series>.min ()</code>	Returns the smallest value.
<code><series>.mode()</code>	Returns the mode for all the values in column named <code><column></code> . Mode is the value that appears most frequently in <code><column></code> .
<code><series>.std ()</code>	Returns a measure of standard deviation. This is the average distance of the values from the mean, expressed in the same units as the original values.
<code><series>.sum ()</code>	Returns the total of the values.
<code><series>.var ()</code>	Returns a measure of variance. This is how much the group of values differs from the mean of the group. This indicates the spread of the values.
<code><series>.value_counts ()</code>	Returns a series containing counts of unique values.

Time

Although there are many different ways of storing and manipulating date and time values, the PLS uses the number of seconds since January 1, 1970, 00:00:00 Coordinated Universal Time (UTC). This base time is known as the epoch. This format is time-zone independent. Individual time zones or local time can be ignored.

Datetime subprograms

<code>datetime.fromtimestamp (<stamp> timezone.utc)</code>	Returns a Coordinated Universal Time (UTC) datetime object equivalent to <stamp>, an integer. The <code>timezone.utc</code> argument enforces the format.
<code>datetime.now (timezone.utc)</code>	Returns a Coordinated Universal Time (UTC) datetime object equivalent to the current local time. The <code>timezone.utc</code> argument enforces the format.
<code>datetime.strptime (<string>, <format>)</code>	Returns a datetime object equivalent to a <string> described by <format>. Set <format> to "%Y-%m-%d %H:%M:%S" for ISO 8601 compatibility.

Datetime object

<code><obj>.date ()</code>	Returns only the date portion of <obj>.
<code><obj>.replace (tzinfo=timezone.utc)</code>	Returns a datetime object with the timezone adjusted for a Coordinated Universal Time (UTC).
<code><obj>.strftime ('%H:%M:%S')</code>	Returns only the time portion of <obj>, formatted for hours, minutes and seconds.
<code><obj>.timestamp ()</code>	Returns a timestamp, which is a real number, representing the date and time in <obj>.

Time

<code>time.sleep (<sec>)</code>	The current process is suspended for the given number of seconds, then resumes at the next line of the program.
<code>time.time ()</code>	Returns a floating-point number which is the current time in seconds since the epoch.

Clean code

Line continuations

Method	Characters	Comment
Inside matched delimiters	Delimiters are: • Round brackets () • Square brackets [] • Curly braces { }	Very easy to read. Adding additional delimiters to allow line breaking at sensible locations is good practice.
Continuation character	The continuation character is: • \	Difficult to read

Comments

Anything on a line after the character # is considered a comment.

Additional programming paradigms

Object-oriented

<code>class <class></code>	Creates a new definition for <class>.
<code>class <class> (<base>)</code>	Creates a new definition for <class> inheriting <base> class.
<code>super().<method></code>	Invokes a method in the parent class.
<code>_<attribute></code>	Indicates a protected attribute.
<code>__<attribute></code>	Indicates a private attribute.
<code><obj>.<method> (<params>)</code>	Invokes the <method> of the instance <obj> with the parameters <params>.
<code><obj>.<attribute></code>	Accesses the <attribute> of the instance <obj>.

Functional

<code>lambda <params> : <expression></code>	Creates an anonymous, inline function with parameters <params> that consists of the code in <expression>.
<code><generator> = <func> (<args>)</code>	Assigns the output of <func> to <generator>. <func> may or may not need <args> passed to it.
<code>next (<generator>)</code>	Retrieves the next value from a lazy evaluation <generator> function.
<code>yield (<value>)</code>	Used in lazy evaluation generator functions to return one value at a time from a sequence.
<code>filter (<func>, <obj>)</code>	Applies <func> to each item in <obj> to determine if the item should be included in the returned result.
<code>map (<func>, <obj>)</code>	Executes <func> for each item in <obj>.
<code>from functools import reduce reduce (<func>, <obj>)</code>	Applies <func> to each item in <obj>, in turn, to produce a single returned result.
<code>zip (<obj1>, <obj2>, ...)</code>	Items from each <obj> are grouped together based on position, i.e. first items together, second items together, etc. The shortest <obj> determines the number of items in the result. The result is an object made up of tuples.

About Pearson

We are the world's leading learning company operating in countries all around the world. We provide content, assessment and digital services to learners, educational institutions, employers, governments, and other partners globally. We are committed to helping equip learners with the skills they need to enhance their employability prospects and to succeed in the changing world of work. We believe that wherever learning flourishes so do people.

References to third party material made in these sample assessment materials are made in good faith. Pearson does not endorse, approve or accept responsibility for the content of materials, which may be subject to change, or any opinions expressed therein. (Material may include textbooks, journals, magazines and other publications and websites.)

All information in this document is correct at time of publication.

Publication code: GQ000052

All the material in this publication is copyright
© Pearson Education Limited 2026

For information about Pearson Qualifications, including Pearson Edexcel and BTEC qualifications, visit qualifications.pearson.com

Edexcel and BTEC are registered trademarks of Pearson Education Limited

Pearson Education Limited. Registered in England and Wales No. 872828
Registered Office: 80 Strand, London WC2R 0RL

VAT Reg No GB 278 537121

Cover image: Shutterstock / archy13