

International Advanced Level **Computer Science**

Sample Assessment Materials

Pearson Edexcel International Advanced Subsidiary in Computer Science (XCP01)

First teaching September 2026 | First examination June 2027

First certification from August 2027

Issue 1

Content

Introduction	1
Unit 1: Principles of computer science	2
Question paper	2
Mark scheme	25
Unit 2: Practical programming and problem-solving	49
Question paper	49
Mark scheme	60
Programming Language Subset (PLS)	97



Introduction

The Pearson Edexcel International Advanced Subsidiary in Computer Science consists of two IAS Units, 1 and 2 and it is the first half of the International Advanced Level qualification. 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.

Onscreen sample assessment material for Unit 1 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 2 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 **WCP01/01**

Computer Science
International Advanced Subsidiary
Unit 1: Principles of Computer Science

You do not need any other materials.

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.

S85794A

©2026 Pearson Education Ltd.
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

Networks and encryption

1 Which **two** statements are characteristics of IPv4 addresses?

- ☐ **A** They are 128-bits in length and written in hexadecimal.
- ☐ **B** They are physical addresses that identify the location of a device.
- ☐ **C** They are written as four groups of numbers separated in dotted-decimal format.
- ☐ **D** They uniquely identify the host interface.
- ☐ **E** They uniquely identify the manufacturer of a device.

(Total for Question 1 = 2 marks)

2 Give the unit of measurement of latency.

.....

.....

(Total for Question 2 = 1 mark)

3 Define the term 'asymmetric encryption'.

.....

.....

.....

.....

(Total for Question 3 = 2 marks)

4 A cipher key for a monoalphabetic substitution cipher is:

W E A K B C D F G H I J L M N O P Q R S T U V X Y Z

The plaintext to encrypt is:

BINARY

Complete the table to show the ciphertext produced using the cipher key.

Table

--	--	--	--	--	--

(Total for Question 4 = 2 marks)

5 Explain **one** way a monoalphabetic substitution cipher is vulnerable to decryption by brute force.

.....

.....

.....

.....

.....

.....

(Total for Question 5 = 2 marks)

6 Explain **one** characteristic of a Vernam cipher that makes it unbreakable.

.....

.....

.....

.....

.....

.....

(Total for Question 6 = 2 marks)

TOTAL FOR SECTION A = 11 MARKS

SECTION B

Structuring data

7 Give **two** types of static data structure.

1

2

(Total for Question 7 = 2 marks)

8 The program code is executed on a dictionary data structure named `cities`.

Program code

```
1 del cities["Berlin"]
2 cities["London"] = 12
3 cities["Shanghai"] = cities["Singapore"]
```

The dictionary `cities` initially contains four key-value pairs.

Cities

Berlin	9
London	11
Singapore	6
Delhi	10

Complete the table to show the updated key-value pairs in the dictionary `cities`.

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

You may not need to use all the rows and cells.

Table

(Total for Question 8 = 3 marks)

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

- 9 A request is made to delete a key-value pair from a dictionary.

Describe the process of searching the dictionary for the key-value pair.

(Total for Question 9 = 2 marks)

- 10 Explain **one** way referential integrity is maintained in a database.

(Total for Question 10 = 2 marks)

TOTAL FOR SECTION B = 9 MARKS

SECTION C**Data representation**

11 Convert the 8-bit binary value 0111 1011 to hexadecimal.

(Total for Question 11 = 2 marks)

12 Give **two** characteristics of the UTF-8 character encoding system.

1

2

(Total for Question 12 = 2 marks)

13 Describe **one** process that could cause an underflow error.

(Total for Question 13 = 2 marks)

14 Signed numbers can be represented in a fixed-point binary format.

The 10-bit signed number is in fixed-point binary format.

Signed number

0	1	0	1	1	0	•	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---

Convert the signed number to denary.

Show your workings.

Answer

(Total for Question 14 = 2 marks)

15 Computers only use binary addition to carry out subtraction.

Calculate the result of the expression:

$$-44 - 11$$

Work in two's complement 8-bit binary throughout.

Show your workings.

Answer

(Total for Question 15 = 6 marks)

TOTAL FOR SECTION C = 14 MARKS

SECTION D

Enabling technologies

16 The five Vs are characteristics that make big data different to other data.

Give **two** of the five Vs of big data.

1

2

(Total for Question 16 = 2 marks)

17 Large language models (LLM) are trained using large data sets.

Explain **one** way bias can be avoided when training a large language model.

.....

.....

.....

.....

.....

.....

.....

(Total for Question 17 = 2 marks)

18 Explain **one** benefit of using unsupervised learning to train a machine learning (ML) model.

.....

.....

.....

.....

.....

.....

.....

(Total for Question 18 = 2 marks)

TOTAL FOR SECTION D = 6 MARKS

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

SECTION E**Problem solving**

19 The truth table shows the output for a Boolean operator.

Truth table

Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	0

Which is the correct Boolean operator for the truth table?

- ☒ **A** AND
- ☒ **B** NAND
- ☒ **C** NOR
- ☒ **D** XOR

(Total for Question 19 = 1 mark)

20 Define the term 'algorithm'.

.....

.....

(Total for Question 20 = 1 mark)

21 A binary search algorithm requires the data in an array to be sorted into order.

Give the name of **one** algorithm that should be used to find the position of an element in an unsorted array.

.....

.....

(Total for Question 21 = 1 mark)

22 Which statement is always true of the procedural programming paradigm?

- ☐ A It uses a low level of abstraction.
- ☐ B It uses imperative commands.
- ☐ C It uses logical schemas.
- ☐ D It uses no flow control.

(Total for Question 22 = 1 mark)

23 Explain **one** benefit of using the declarative programming paradigm.

.....

.....

.....

.....

.....

.....

(Total for Question 23 = 2 marks)

24 The algorithm calculates the factorial of a number.

Algorithm

```
1 def factorial(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * factorial(n-1)
```

Give the name of the problem-solving technique used by the algorithm.

.....

.....

(Total for Question 24 = 1 mark)

25 The algorithm outputs how many times a target value appears in an array.

Algorithm

```
1 times = [14, 8, 6, 14, 20, 10, 14, 20]
2 search_target = 14
3 index = 0
4 count = 0
5
6 while index < 8:
7     if times[index] == search_target:
8         count = count + 1
9     index = index + 1
10
11 print("The target appears", count, "times")
```

An error occurs if additional items are added to the array in line 1 of the algorithm.

Explain the type of error that occurs.

.....

.....

.....

.....

.....

.....

(Total for Question 25 = 2 marks)

26 The program is for a binary search.

Program

```
1 distances = [1.2, 1.6, 2.11, 2.8, 3.08, 3.47, 3.61, 3.7, 3.93, 4.22, 4.85]
2 lower = 0
3 upper = len(distances) - 1
4 mid = 0
5 found = False
6
7 target = float(input("Please enter the number you want to search for "))
8
9 while index < 8:
10     mid = (lower + upper)//2
11     if distances[mid] == target:
12         found = True
13     elif distances[mid] < target:
14         lower = mid + 1
15     else:
16         upper = mid - 1
17
18 if found:
19     print("{} was found at position {}".format(target, mid))
20 else:
21     print("{} does not appear in the data".format(target))
```

Complete the table using the program for the binary search to:

- trace the execution of the program with the input data 3.7
- give the output of the program.

You may not need to use all the rows and cells.

Table

lower	upper	mid	distances [mid]	found	target
Program output					

(Total for Question 26 = 4 marks)

27 A car has a warning system. The warning system sounds an alarm if a sensor detects at least one of these conditions:

- there is an object behind the car when the car is reversing
- there is a vehicle in the lane the car is indicating a move to.

The symbols used to represent the warning system are:

Warning system symbols

Symbol	True when
Q	the alarm sounds
A	there are no objects behind the car
B	the car is reversing
C	there is no vehicle in the lane the car is moving to
D	the car is indicating

Construct a Boolean expression for the warning system using the symbols.

.....

.....

.....

.....

.....

.....

.....

.....

(Total for Question 27 = 4 marks)

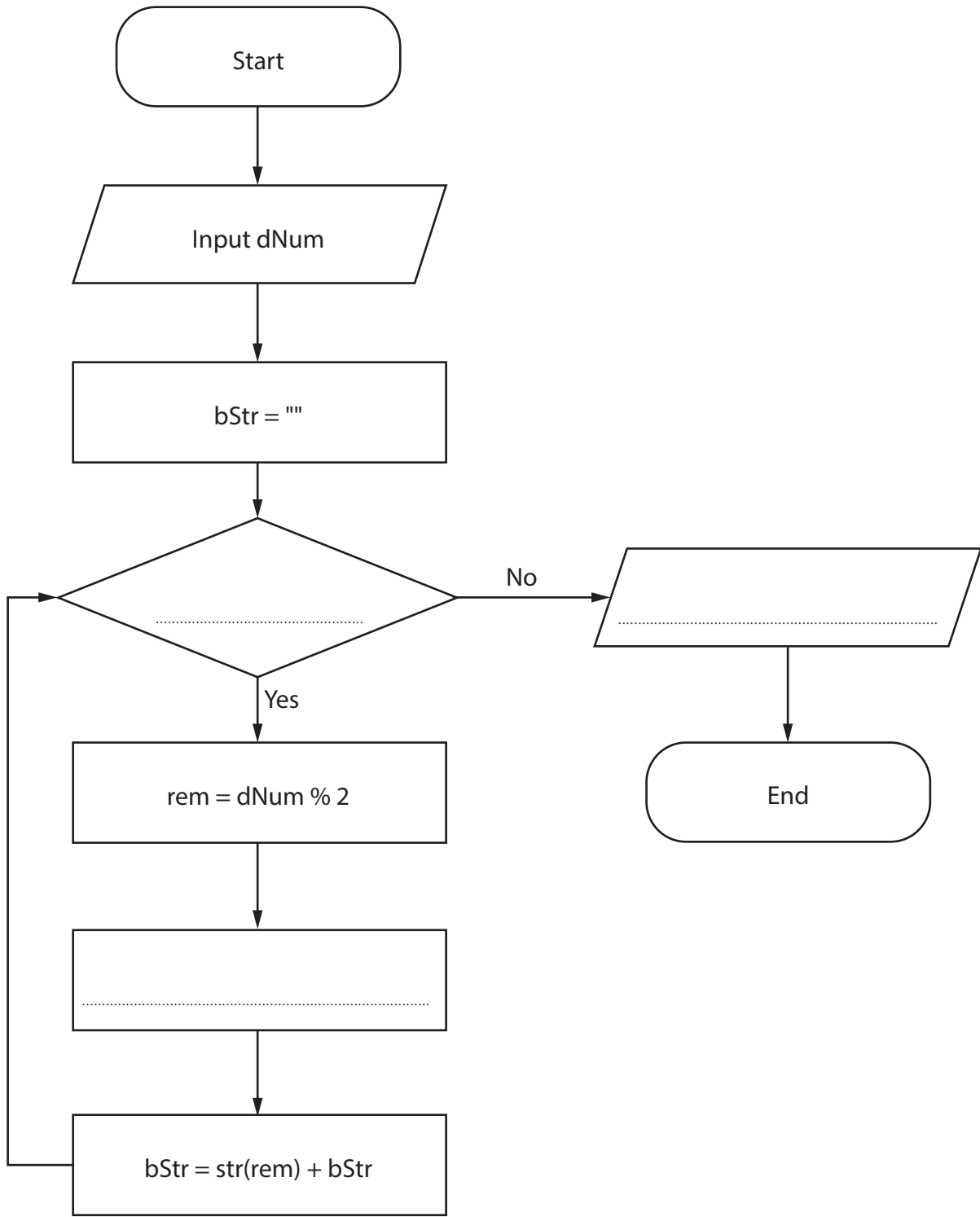
28 The partially completed flowchart is for an algorithm. The algorithm converts a denary integer into its binary value using repeated division. It outputs the result as a string.

The variables used are:

- dNum stores the positive denary integer
- bStr stores the binary string
- rem stores the remainder after each division.

Complete the flowchart by adding the **three** missing operations.

Flowchart



(Total for Question 28 = 3 marks)

TOTAL FOR SECTION E = 20 MARKS

DO NOT WRITE IN THIS AREA

SECTION F

Computer systems

29 Give **two** benefits of the stored-program concept.

- 1
- 2

(Total for Question 29 = 2 marks)

30 Two purposes of an operating system (OS) are memory management and multitasking.

Give **two other** purposes of an operating system.

- 1
- 2

(Total for Question 30 = 2 marks)

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

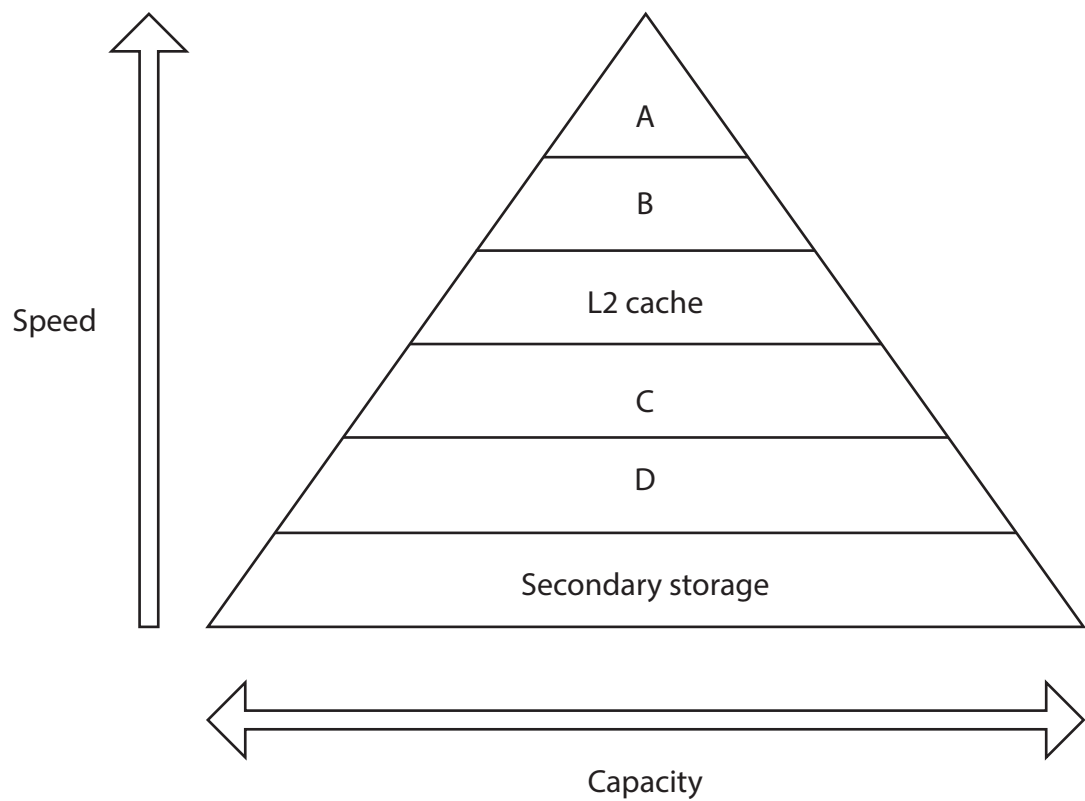
DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

31 In a laptop, the memory holds executing processes and their data.

The labelled diagram shows the memory hierarchy. It is arranged in order of increasing speed and decreasing capacity.

Memory hierarchy



Complete the table by adding the type of memory for the **four** labels in the memory hierarchy.

Table

Label	Type of memory
A	
B	
C	
D	

(Total for Question 31 = 2 marks)

32 The table shows some of the steps in handling an interrupt received as part of multitasking.

Complete the table by adding the **two** missing steps.

Table

1	Suspends the currently executing process
2	
3	Services the interrupt
4	
5	Resumes executing the suspended process

(Total for Question 32 = 2 marks)

33 Errors can occur when data is transmitted.

Explain **one** reason skew only affects parallel data transmission.

.....

.....

.....

.....

.....

.....

(Total for Question 33 = 2 marks)

34 An operating system (OS) allocates and deallocates memory as part of memory management.

Outline how segmentation is used in memory management.

(Total for Question 34 = 2 marks)

35 One component of the central processing unit (CPU) is the memory buffer register (MBR).

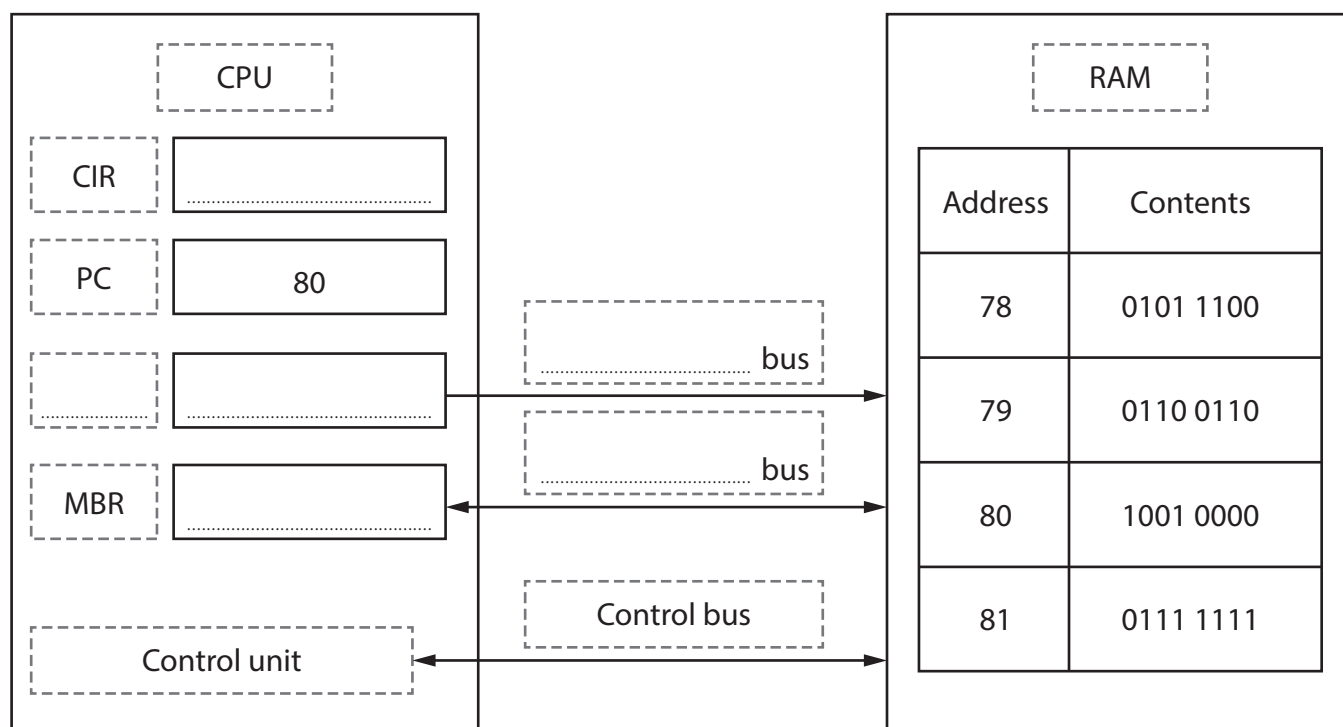
Explain **one** reason the MBR is needed.

(Total for Question 35 = 2 marks)

- 36** The partially completed diagram shows the components of a computer system. The components are used in the fetch-decode-execute (FDE) cycle.

Complete the diagram by adding the **six** missing items to show the next instruction being fetched.

Diagram



(Total for Question 36 = 6 marks)

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

International Advanced Subsidiary in Computer Science

Unit 1: 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 two correct answers are C (1) and D (1) A is not correct because it applies to IPv6 addresses B is not correct because IPv4 addresses are logical addresses E is not correct because it applies to MAC addresses	The correct answers are: C They are written as four groups of numbers separated in dotted-decimal format. D They uniquely identify the host interface.	2

Question number	Answer	Additional Guidance	Mark
2	Award one mark for: • milliseconds(1)	Accept: ms s/seconds	1

Question number	Answer	Additional Guidance	Mark
3	Award up to two marks for a linked definition that makes reference to: • Uses two keys / key pair / public and private key (1) one key for encryption and a different key for decryption (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark						
4	Award one mark for each of the following up to a maximum of two marks: - Any three letters correct (1) - Remaining three letters correct (1) <table><tr><td>E</td><td>G</td><td>M</td><td>W</td><td>Q</td><td>Y</td></tr></table>	E	G	M	W	Q	Y	Ignore case	2
E	G	M	W	Q	Y				

Question number	Answer	Additional Guidance	Mark
5	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks: - Frequency analysis can reveal patterns in the ciphertext (1) because each plaintext letter is always encrypted to the same ciphertext letter (1) - An exhaustive search will decrypt the message (1) because the number of possible keys is limited/26 factorial (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
6	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks: • The key is genuinely random (1) so that patterns will not emerge in the ciphertext (1) • The key is only used once (1) so that multiple messages cannot be compared to reveal information about the key (1) • The key is at least as long as the plaintext/message (1) since repeating the key (to match the length of the plaintext/message) would introduce patterns into the ciphertext (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
7	Award one mark for each of the following up to a maximum of two marks: • Array (1) • Tuple (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark												
8	Award one mark for each of the following up to a maximum of three marks: • Berlin element deleted (1) • Existing London value amended to 12 (must not be as an additional element) (1) • Additional element added with key of Shanghai and value of 6 (1) Maximum two marks if any other changes are seen. Exemplar completed table: <table><tr><td>London</td><td>12</td></tr><tr><td>Singapore</td><td>6</td></tr><tr><td>Delhi</td><td>10</td></tr><tr><td>Shanghai</td><td>6</td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>	London	12	Singapore	6	Delhi	10	Shanghai	6					Rows can be listed in any order as long as the values are paired with correct keys. Ignore minor spelling errors in city names and ignore empty rows.	3
London	12														
Singapore	6														
Delhi	10														
Shanghai	6														

Question number	Answer	Additional Guidance	Mark
9	Award one mark for each appropriate point given in the correct order, up to a maximum of two marks: The key will be hashed (to find the position/index) (1) and the key-value paid at the position/index will be deleted (1). Accept any other appropriate phrasing.		2

Question number	Answer	Additional Guidance	Mark
10	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks. - Each table must have a unique primary key (1) so that any foreign key in a related table matches an existing primary key (1) - Deleting a record from the parent table should trigger deletions from child tables (1) so that orphaned records cannot occur (1) - Updates in the parent table should be cascaded to child tables (1) so that inconsistent data is prevented (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
11	Award two marks for a fully correct conversion to hexadecimal: • 7B (2) OR Award one mark for either of the following in the correct position: • 7 (1) • B (1)	Ignore case and blank spaces.	2

Question number	Answer	Additional Guidance	Mark
12	Award one mark for any of the following, up to a maximum of two marks: • It can represent many different writing systems/emojis/symbols (1) • It uses a variable-width encoding system / It uses between 1 and 4 bytes per character (1) • It enables backward compatibility with ASCII (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
13	Award one mark for each appropriate point given in the correct order, up to a maximum of two marks: • A numerical calculation/binary shift is performed by the ALU/processor (1) and the result is too small to be (accurately) represented/stored (in the bits available) (1) • A number of small magnitude is divided by a number of large magnitude (1) resulting in a number too small to be (accurately) represented (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
14	Award two marks for a fully correct conversion to denary: • $22\frac{5}{8} / 22.625$ (2) OR Award one mark for either of the following: • 22 (1) • $5/8 / \frac{5}{8} / .625$ (1)	If answering in fractions, answers must clearly show $22\frac{5}{8}$ not $225/8$	2

Question number	Answer	Additional Guidance	Mark																																				
15	Award one mark for each of the following, up to a maximum of six marks: • Convert -44 to signed binary 1010100 (1) • Pad the result for -44 with 1(s) on the left up to 8 bits (1) • Convert -11 to signed binary 10101 (1) • Pad the result for -11 with 1(s) on the left up to 8 bits (1) • Evidence of adding in binary with carries (1) • Fully correct answer (1) Exemplar full method: Convert both values to signed binary and pad: $-44_{10} = (1)1010100_2$ $-11_{10} = (111)10101_2$ Add values together: Working like this may be seen (with carry bits recorded below or above) <table><tr><td>-44</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>-11</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>Answer</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>Carry</td><td>1</td><td>1</td><td>1</td><td>1</td><td></td><td></td><td></td><td></td></tr></table>	-44	1	1	0	1	0	1	0	0	-11	1	1	1	1	0	1	0	1	Answer	1	1	0	0	1	0	0	1	Carry	1	1	1	1					If more than 8-bits given in the answer, do not award mp6. Allow other equivalent approaches for working.	6
-44	1	1	0	1	0	1	0	0																															
-11	1	1	1	1	0	1	0	1																															
Answer	1	1	0	0	1	0	0	1																															
Carry	1	1	1	1																																			

Question number	Answer	Additional Guidance	Mark
16	Award one mark for any of the following, up to a maximum of two marks: • Variety (1) • Velocity (1) • Veracity (1) • Volume (1) • Value (1)		2

Question number	Answer	Additional Guidance	Mark
17	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks: • The training data must be diverse/representative / must be designed to actively include under-represented groups (1) because the LLM cannot produce representative output without having learnt from diverse sources (1) • The training data must be filtered/pre-processed to remove explicitly biased material (1) so that the LLM does not reproduce it (1) • The training data must include the context of the model (1) so that the model does not make judgements based on inaccurate norms for the scenario/time period/location (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
18	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks: - The model may be able to identify clusters/associations (1) because it identifies naturally occurring patterns in data sets (1) - Training can begin more quickly (1) because it does not need training data that is already labelled/categorised (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
19	The correct answer is D A is not correct because the output Q would be 0001 for AND B is not correct because the output Q would be 1110 for NAND C is not correct because the output Q would be 1000 for NOR	The correct answer is D XOR	1

Question number	Answer	Additional Guidance	Mark
20	Award one mark for a definition that makes reference to: It is a finite sequence of instructions/rules/steps to solve a problem/complete a task (1) Accept any other appropriate response.		1

Question number	Answer	Additional Guidance	Mark
21	Award one mark for: Linear search (1) Accept any other appropriate response.		1

Question number	Answer	Additional Guidance	Mark
22	The correct answer is B A is not correct because the procedural programming paradigm uses a high level of abstraction C is not correct because the procedural programming paradigm does not use logical schemas. D is not correct because the procedural programming paradigm does use flow control.	The correct answer is B It uses imperative commands.	1

Question number	Answer	Additional Guidance	Mark
23	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks: • It reduces complexity (1) because the programmer says what they want to happen rather than how it happens (1) • It improves readability/maintainability (1) because there are fewer instructions overall (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
24	Award one mark for: • Recursion(1)	Do not accept function.	1

Question number	Answer	Additional Guidance	Mark
25	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks: • A logic error occurs (1) because not all the items in the list will be searched / only the first 8 items will be searched / the loop condition is hard-coded (1) Accept any other appropriate response.	Do not accept generic explanation of a logic error.	2

Question number	Answer	Additional Guidance	Mark																																																																																										
26	Award one mark for each of the following up to a maximum of four marks: • First row correct (1) • First iteration correct (1) • Remaining iterations correct (1) • found set to True on last row and output “3.7 was found at position 7” (1) Exemplar completed table: <table><tr><th>lower</th><th>upper</th><th>mid</th><th>distances[mid]</th><th>found</th><th>target</th></tr><tr><td>0</td><td>10</td><td>0</td><td></td><td>False</td><td>3.7</td></tr><tr><td></td><td></td><td>5</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td>3.47</td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td>8</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td>3.93</td><td></td><td></td></tr><tr><td></td><td>7</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td>6</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td>3.61</td><td></td><td></td></tr><tr><td>7</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td>7</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td>3.7</td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>True</td><td></td></tr><tr><td colspan="2">Program output</td><td colspan="4">3.7 was found at position 7</td></tr></table> Students may choose to write one iteration on each row – accept so long as the same values are present.	lower	upper	mid	distances[mid]	found	target	0	10	0		False	3.7			5							3.47			6								8							3.93				7							6							3.61			7								7							3.7							True		Program output		3.7 was found at position 7				Ignore minor spelling errors in the output. Ignore 1.2 in distances [mid] column in the first row. Mark points in the table are colour coded: BP1 BP2 BP3 BP4	4
lower	upper	mid	distances[mid]	found	target																																																																																								
0	10	0		False	3.7																																																																																								
		5																																																																																											
			3.47																																																																																										
6																																																																																													
		8																																																																																											
			3.93																																																																																										
	7																																																																																												
		6																																																																																											
			3.61																																																																																										
7																																																																																													
		7																																																																																											
			3.7																																																																																										
				True																																																																																									
Program output		3.7 was found at position 7																																																																																											

Question number	Answer	Additional Guidance	Mark																																																																																																
26 (continued)	Alternative exemplar table: <table><tr><th>lower</th><th>upper</th><th>mid</th><th>distances[mid]</th><th>found</th><th>target</th></tr><tr><td>0</td><td>10</td><td>0</td><td></td><td>False</td><td>3.7</td></tr><tr><td></td><td></td><td>5</td><td>3.47</td><td></td><td></td></tr><tr><td>6</td><td></td><td>8</td><td>3.93</td><td></td><td></td></tr><tr><td></td><td>7</td><td>6</td><td>3.61</td><td></td><td></td></tr><tr><td>7</td><td></td><td>7</td><td>3.7</td><td>True</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td colspan="2">Program output</td><td colspan="4">3.7 was found at position 7</td></tr></table> Students may insert values in every cell by repeating unchanging variables – accept. Alternative exemplar table: <table><tr><th>lower</th><th>upper</th><th>mid</th><th>distances[mid]</th><th>found</th><th>target</th></tr><tr><td>0</td><td>10</td><td>0</td><td></td><td>False</td><td>3.7</td></tr><tr><td>0</td><td>10</td><td>5</td><td>3.47</td><td>False</td><td>3.7</td></tr><tr><td>6</td><td>10</td><td>8</td><td>3.93</td><td>False</td><td>3.7</td></tr><tr><td>6</td><td>7</td><td>6</td><td>3.61</td><td>False</td><td>3.7</td></tr><tr><td>7</td><td>7</td><td>7</td><td>3.7</td><td>True</td><td>3.7</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td colspan="2">Program output</td><td colspan="4">3.7 was found at position 7</td></tr></table>	lower	upper	mid	distances[mid]	found	target	0	10	0		False	3.7			5	3.47			6		8	3.93				7	6	3.61			7		7	3.7	True								Program output		3.7 was found at position 7				lower	upper	mid	distances[mid]	found	target	0	10	0		False	3.7	0	10	5	3.47	False	3.7	6	10	8	3.93	False	3.7	6	7	6	3.61	False	3.7	7	7	7	3.7	True	3.7							Program output		3.7 was found at position 7					
lower	upper	mid	distances[mid]	found	target																																																																																														
0	10	0		False	3.7																																																																																														
		5	3.47																																																																																																
6		8	3.93																																																																																																
	7	6	3.61																																																																																																
7		7	3.7	True																																																																																															
Program output		3.7 was found at position 7																																																																																																	
lower	upper	mid	distances[mid]	found	target																																																																																														
0	10	0		False	3.7																																																																																														
0	10	5	3.47	False	3.7																																																																																														
6	10	8	3.93	False	3.7																																																																																														
6	7	6	3.61	False	3.7																																																																																														
7	7	7	3.7	True	3.7																																																																																														
Program output		3.7 was found at position 7																																																																																																	

Question number	Answer	Additional Guidance	Mark
27	Award four marks for a fully correct expression: • $Q = \bar{A}.B + \bar{C}.D$ (4) OR Award one mark for any of the following up to maximum of three marks: • + between two expressions (1) • $\bar{A}.B$ (1) • $\bar{C}.D$ (1)		4

Question number	Answer	Additional Guidance	Mark
28	Award one mark for each correct operation in the correct box, up to a maximum of three marks: • <code>dNum > 0</code> (1) • <code>dNum = dNum // 2</code> (1) • Output <code>bStr</code> (1) Exemplar answer: <pre>graph TD Start([Start]) --> Input[/Input dNum/] Input --> Init[bStr = ''] Init --> Decision{Is dNum > 0?} Decision -- Yes --> Rem[rem = dNum % 2] Rem --> Div[dNum = dNum // 2] Div --> Append[bStr = str(rem) + bStr] Append --> Decision Decision -- No --> Print[/Print bStr/] Print --> End([End])</pre> Accept any other appropriate response.	Accept <code>dNum != 0</code> Accept <code>dNum = dNum div 2</code> Accept alternative syntax e.g. <code>dNum <> 0</code> / <code>print(bStr)</code>	3

Question number	Answer	Additional Guidance	Mark
29	Award one mark for each of the following, up to a maximum of two marks: • Allows computers to execute different programs/tasks easily (without the need to reconfigure hardware) (1) • The computer can modify its own data or program (1) • Fast execution (as both programs and data are in the same memory) (1) Accept any other appropriate response.		2

Question number	Answer	Additional Guidance	Mark
30	Award one mark for each of the following, up to a maximum of two marks: • Peripheral/device management (1) • User interface (1) • User management (1) • Process management (1) Accept any other appropriate response.	Do not accept multitasking and memory management.	2

Question number	Answer	Additional Guidance	Mark										
31	Award one mark for each of the following up to a maximum of two marks: • CPU registers and main memory/RAM in the correct positions (1) • L1 and L3 cache in the correct position (1) Exemplar answer: <table><tr><th>Label</th><th>Type of Memory</th></tr><tr><td>A</td><td>CPU registers</td></tr><tr><td>B</td><td>L1 cache</td></tr><tr><td>C</td><td>L3 cache</td></tr><tr><td>D</td><td>Main Memory/RAM</td></tr></table>	Label	Type of Memory	A	CPU registers	B	L1 cache	C	L3 cache	D	Main Memory/RAM		2
Label	Type of Memory												
A	CPU registers												
B	L1 cache												
C	L3 cache												
D	Main Memory/RAM												

Question number	Answer	Additional Guidance	Mark					
32	Award up to two marks for completed boxes: <table><tr><td>1. Suspends the currently executing process</td></tr><tr><td>2. Status is saved in a stack frame (1)</td></tr><tr><td>3. Services the interrupt</td></tr><tr><td>4. Status is popped from the stack frame (1)</td></tr><tr><td>5. Resumes executing the suspended process</td></tr></table> Accept alternative wording for the steps.	1. Suspends the currently executing process	2. Status is saved in a stack frame (1)	3. Services the interrupt	4. Status is popped from the stack frame (1)	5. Resumes executing the suspended process		2
1. Suspends the currently executing process								
2. Status is saved in a stack frame (1)								
3. Services the interrupt								
4. Status is popped from the stack frame (1)								
5. Resumes executing the suspended process								

Question number	Answer	Additional Guidance	Mark
33	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks: Bits from the same word can get out of step with each other (1) because they are transmitted along multiple wires at the same time (1) Accept any other appropriate response.	Accept explanation of why skew cannot affect serial transmission	2

Question number	Answer	Additional Guidance	Mark
34	Award one mark for each appropriate point in a linked response up to a maximum of two marks: • (Logical) memory is split into variable-sized blocks/partitions/segments (1), which are loaded into physical memory (for execution) (1) • (Logical) memory is split into variable-sized blocks/partitions/segments (1). The segment table maps the segment to physical memory locations (1) Accept any other appropriate response.	Do not accept segments on its own. Answer must refer to variable size.	2

Question number	Answer	Additional Guidance	Mark
35	Award one mark for an identification point and one mark for an appropriate linked justification, up to a maximum of two marks: • It makes up for the difference in speed between the CPU and RAM (1) by providing a temporary data store (1) • It enables the CPU to handle more instructions per second (1) because it frees up the data bus to be used again for another fetch (1) • It acts as a buffer between the different speeds of the CPU and the RAM (1) because it stores data until the data bus is ready (1) Accept any other appropriate response.	Accept MDR	2

Question number	Answer	Additional Guidance	Mark
36	Award one mark for each of the following up to a maximum of six marks: • Address bus (1) • Data bus (1) • MAR (1) • 80 (1) • 10010000 in MBR (1) • 10010000 in CIR (1) Exemplar answer: The diagram illustrates a computer system with two main components: a CPU and RAM. The CPU is represented by a large rectangle containing several sub-components: a dashed box labeled 'CPU' at the top, a 'CIR' (Current Instruction Register) containing the value '1001 0000', a 'PC' (Program Counter) containing the value '80', a 'MAR' (Memory Address Register) containing the value '80', an 'MBR' (Memory Buffer Register) containing the value '1001 0000', and a 'Control Unit' at the bottom. The RAM is represented by a large rectangle containing a dashed box labeled 'RAM' at the top and a table of memory locations below. The table has two columns: 'Address' and 'Contents'. The rows are: Address 78 with Contents '0101 1100', Address 79 with Contents '0110 0110', Address 80 with Contents '1001 0000', and Address 81 with Contents '0111 1111'. Three buses connect the CPU and RAM: an 'Address bus' (dashed box) connects the PC and MAR to the RAM's Address column; a 'Data bus' (dashed box) connects the MBR to the RAM's Contents column; and a 'Control bus' (dashed box) connects the Control Unit to the RAM.		6

Pearson Edexcel International Advanced Level

Sample assessment material

Time 3 hours

Paper
reference

WCP02/01

Computer Science

International Advanced Subsidiary Level

Unit 2: 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 Programming Language Subset (PLS) document.

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 if you have time at the end.
- Try to answer every question.

S85795A

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



Answer ALL questions.

Suggested time: 20 minutes

- 1** A program is intended to implement and test a stack.

The program runs three logical tests. The output from the tests helps to debug the errors.

Open file **Q01.py**

Amend the code to:

- fix the syntax error on original line 10
`my_stack = [None, None, None, None None, None, None, None, None, None]`
- fix the syntax error on original line 33
`def is_empty()`
- fix the syntax error on original line 45
`status =! True`
- fix the syntax error on original line 94
`push ("Banana)`
- fix the TypeError on original line 20
`for index in range(STACK_SIZE * 0.5)`
- fix the TypeError on original line 29 by using type conversion
`out_string = out_string + item + " "`
- fix the UnboundLocalError on original line 62
`top_stack`
- fix the NameError on original line 91
`clean_stack()`
- fix the TypeError on original line 95
`show_stack("Apple")`
- fix the logic error on original line 23 that reports the wrong index for the top of the stack
`top_stack = -9`
- fix the logic error on original line 65 that pops None, in error, when the stack is empty
`if(is_empty()):`
- fix the logic error on original line 54 that skips stack spaces when pushing
`top_stack = top_stack + 2`

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 palindrome is a string that reads the same forwards and backwards.

A program is intended to determine if a string is a palindrome.

The program loops until the user enters an empty string.

The algorithm in the program must convert uppercase letters to lowercase letters and remove punctuation and symbols before processing.

The test data is for a fully-functional program.

Test data

Input	Output
civic	Original: civic Prepared: civic is a palindrome
CiVic	Original: CiVic Prepared: civic is a palindrome
No melon, no lemon	Original: No melon, no lemon Prepared: nomelonnolemon is a palindrome
Coffee	Original: Coffee Prepared: coffee is not a palindrome
34\$543	Original: 34\$543 Prepared: 34543 is a palindrome

The program does not work.

Some lines are commented out.

Some lines are not in the correct location.

All lines are correctly indented.

Open file **Q02.py**

Amend the code to produce the correct output.

You will need to:

- rearrange some lines
- choose the correct line of code. Make a choice by removing the # at the beginning of the lines you choose to execute.

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 = 15 marks)

Suggested time: 40 minutes

- 3 A company sells products to customers. The products are supplied by makers. Makers specialise in different crafts. Specialisms include woodworking and 3D printing.

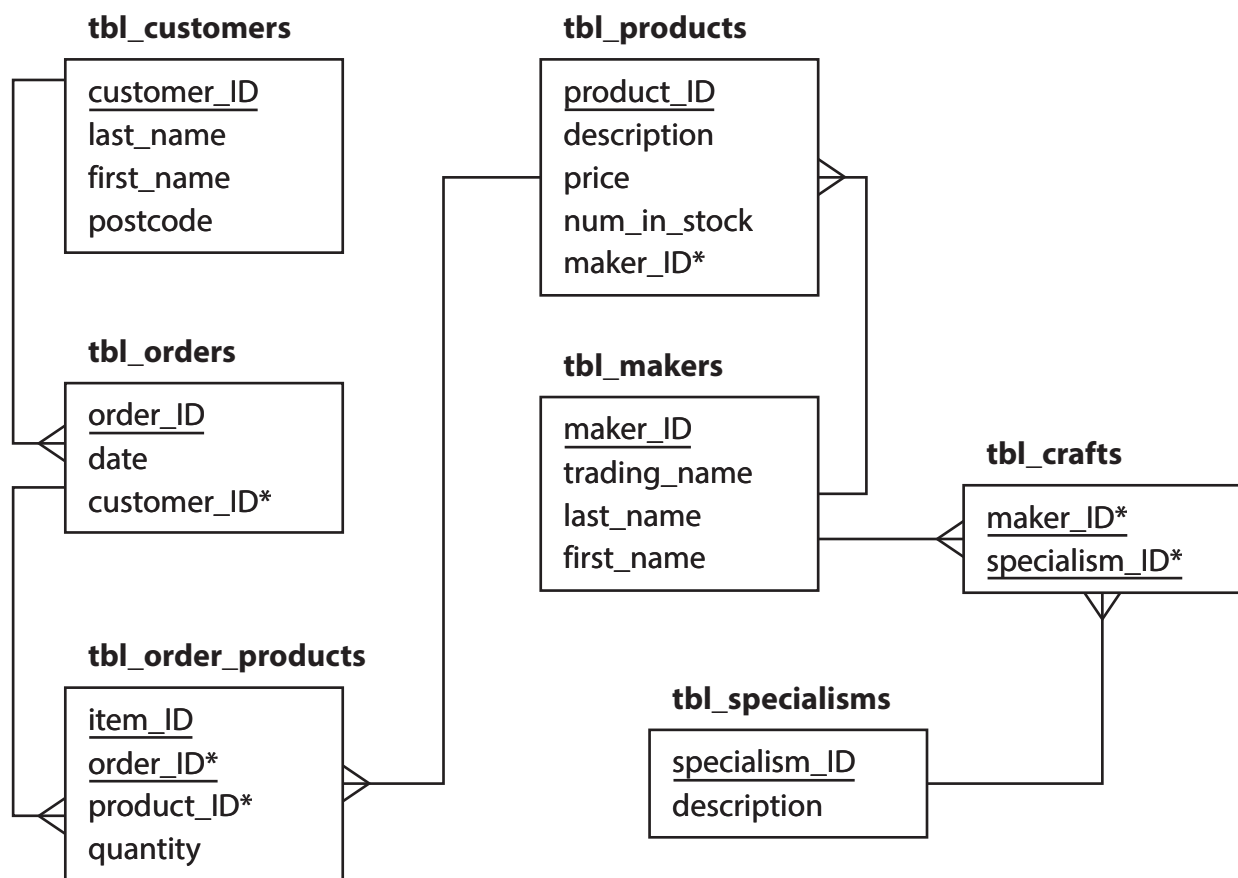
The entity-relationship diagram is for the existing CustomCrafts database.

Table names are emboldened.

Primary keys are underlined.

Foreign keys are indicated by *

Entity-relationship



The CustomCrafts database is supplied in the file CustomCrafts.db

An employee table is defined by:

tbl_employees (employee_ID, address_1, address_2, postcode)

- employee_ID is an integer
- address_1, address_2, and postcode are text.

The output is from a fully-functional program.

Output

```
There are 12 different products.  
Product 9 has sold 5 times.  
The customers who have postcodes that start with 608087 are:  
(327675, 'Evans', 'Erin', '6080873745')  
(534980, 'Adams', 'Aspen', '6080878902')  
(571526, 'Franks', 'Faye', '6080872932')  
The trading names of the makers who specialise in Woodworking are:  
( 'Natural Materials', )  
( 'Delicate Designs', )
```

Open file **Q03.py**

The code has not implemented any error handling because no error handling is required.

Amend the code to:

- determine how many different products are held in the database
- determine the quantity of product ID 9 that has been sold
- determine all the customers who have postcodes that start with 608087
- add the employee table to the database
- determine the trading names of the makers who specialise in woodworking.

The program must:

- output messages that are fit for purpose
- call the supplied subprogram, `show_result`, to display results that have multiple items.

Use the library 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 = 15 marks)

Suggested time: 40 minutes

- 4 A program is required to sort lists of integers into ascending order.

There are two lists in the program code file. The `little_list` is filled with integers. The `big_list` is empty and must be filled with random integers.

An in-place insertion sort algorithm is required to do the sorting.

The table is an example of using the in-place insertion sort algorithm to sort the five integers in `little_list`.

Table

little_list					Description
44	22	11	33	55	Original
44		11	33	55	22 is copied into a target variable
	44	11	33	55	44 shifts right
22	44	11	33	55	22 is copied into the gap
22	44		33	55	11 is copied into a target variable
	22	44	33	55	44 shifts right; 22 shifts right
11	22	44	33	55	11 is copied into the gap
11	22	44		55	33 is copied into a target variable
11	22		44	55	44 shifts right
11	22	33	44	55	33 is copied into the gap
11	22	33	44		55 is copied into a target variable
11	22	33	44		No shifts required
11	22	33	44	55	55 is copied into the gap
11	22	33	44	55	End of list reached

Open file **Q04.py**

Amend the code to:

- import a required library
- complete the insertion sort procedure that takes a list as a parameter and sorts the list in place. No return value is required.
- manipulate `little_list` to:
 - display the unsorted list (no formatting required)
 - sort the list by calling the insertion sort procedure
 - display the sorted list (no formatting required)
- complete the `display_list_matrix()` procedure to display the list in five rows of 20 integers each
- manipulate `big_list` to:
 - fill the list with 100 random integers between 0 and 100
 - display the unsorted list by calling `display_list_matrix()`
 - sort the list using the insertion sort procedure
 - display the sorted list by calling `display_list_matrix()`.

Use the constant and variables provided.

Execute the code to test its functionality.

Do **not** use a built-in or library sorting subprogram.

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: 60 minutes

5 An air quality monitoring system has a network of sensors placed across a city.

Sensors take readings of the amount of nitrogen dioxide (NO₂) in the air.

Sensors take one reading every 10 minutes, 24 hours a day, seven days a week.

Readings are stored as µg/m³ (micrograms per cubic metre of air).

Readings are reported to one decimal place.

The minimum valid sensor reading is 10.0

The maximum valid sensor reading is 51.9

A sensor is faulty when it reports the first invalid reading.

Engineers will use a program to identify faulty sensors before the sensors fail completely.

Description of the dataset

A sample of the full dataset is provided in the program code file.

Each row of the dataset consists of:

- sensor ID (integer)
- latitude (float)
- longitude (float)
- location (string)
- timestamp (integer, UTC format)
- reading (float).

The output is from a fully-functional program.

The output shows the first time a sensor gives an invalid reading.

Output

There are 4 faulty sensors

ID	Location	Date/Time	Reading
16	Hyde Park Street	2024-11-05 02:50:00	-33.7
18	Abbey Wood	2024-11-01 00:10:00	55.7
47	Greenwich Park	2024-11-05 06:30:00	0.0
125	Green Park	2024-11-03 04:50:00	122.8

Formatting requirements describes the output layout.

Formatting requirements

ID	Location	Date/Time	Reading
Left aligned	Centre aligned	Centre aligned	Right aligned
5 characters	22 characters	20 characters	7 characters

Open file **Q05.py**

Write the code to:

- use the dataset to produce the output shown in output
- format the output to meet the requirements shown in formatting requirements
- display the output to the screen
- write the output to a text file.

The program must:

- work for any number of sensors in the dataset
- include at least one user-defined subprogram that takes parameters.

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 Subsidiary in Computer Science

Unit 2: 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	10	A comma is added after 4 th occurrence of None None, None, None, None, None, None, None, None		12
	1.2	20	Multiplication by 1/5 (* 0.5) is removed from range argument range(STACK_DEPTH):		
	1.3	23	The -9 is changed to -1 top_stack = -1		
	1.4	29	Item is converted from None to string str(item)		
	1.5	33	A semicolon is added to end of definition line def is_empty():		
	1.6	45	The ! symbol is removed from right of = symbol status = True		
	1.7	54	The +2 is changed to +1 top_stack + 1		

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
1 (continued)	1.8	62	The global keyword is added for top_stack global top_stack		12
	1.9	65	Not is added to negate returned value if (not is_empty()):		
	1.10	91	The identifier name is corrected by removing capitalisation clean_stack()		
	1.11	94	A closing closing double quote is added to string "Banana"		
	1.12	95	The argument "Apple" is removed show_stack()		

```

1  # -----
2  # Constants
3  # -----
4  STACK_SIZE = 10          # Number elements that can be held
5
6  # -----
7  # Global variables
8  # -----
9  # Fixed-length list
10 my_stack = [None, None, None, None, None, None, None, None, None, None]
11
12 top_stack = -1           # No items in the list; tracking indexes
13
14 # -----
15 # Subprograms
16 # -----
17 def clean_stack():
18     global top_stack
19
20     for index in range(STACK_SIZE):
21         my_stack[index] = None
22
23     top_stack = -1
24
25 def show_stack():
26     out_string = ""
27
28     for item in my_stack:
29         out_string = out_string + str(item) + " "
30     print(out_string)
31     print("top = " + str(top_stack))
32
33 def is_empty():
34     status = False
35
36     if(top_stack == -1):
37         status = True
38
39     return(status)
40
41 def is_full():
42     status = False
43
44     if(top_stack == STACK_SIZE - 1):
45         status = True
46
47     return(status)
48

```

```

49 def push(p_item):
50     global top_stack
51     status = None
52
53     if(not is_full()):
54         top_stack = top_stack + 1
55         my_stack[top_stack] = p_item
56     else:
57         status = False          # Can't push
58
59     return(status)
60
61 def pop():
62     global top_stack
63     status = "Empty"
64
65     if(not is_empty()):
66         item = my_stack[top_stack]
67         top_stack = top_stack - 1
68         return(item)
69     else:
70         return(status)
71
72 # -----
73 # Main program
74 # -----
75 print("-"*50)
76 print ("Logic Test 1: Before the code is corrected, the top of stack is reported")
77 clean_stack()
78 show_stack()
79
80 # -----
81 print("-"*50)
82 print ("Logic Test 2: Before the code is corrected, the stack is empty so nothing")
83 clean_stack()
84 show_stack()
85 print (pop())
86 show_stack()
87
88 # -----
89 print("-"*50)
90 print ("Logic Test 3: Before the code is corrected, positions in the stack are")
91 clean_stack()
92 show_stack()
93 push("Apple")
94 push("Banana")
95 show_stack()
96

```

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
2	Check for Palindrome			Do not award marks in this section, where the order of the lines matches the original student code	15
	2.1	17	Definition line is the first line in the subprogram <pre>def is_palindrome (p_string):</pre>		
	2.2	18	Length is calculated before any other instruction <pre>length = len(p_string)</pre>		
	2.3	19 21 23	The selection statement keywords are placed in order as: 1. If 2. Elif 3. Else	- Must have at least one instruction indented under each keyword, i.e. no empty subblock - Ignore accuracy of indented code	
	2.4	20	The return (True) is placed inside the if selection subblock <pre>if (length < 2): return (True)</pre>		
	2.5	21 22	The return (recursive call) is placed inside the elif subblock <pre>elif (p_string[0] == p_string[length - 1]): return (is_palindrome(p_string[1:length - 1]))</pre>		

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
2 (continued)	2.6	23 24	The return (False) is placed inside the else subblock else: return (False)		15
	Prepare String				
	2.7	29	The correct line has the # symbol removed def prepare_string(p_string):	2nd in sequence of four	
	2.8	36	The correct line has the # symbol removed for index in range(len(p_string)):	1st in sequence of four	
	2.9	46	The correct line has the # symbol removed if (char.isalnum()):	4th in sequence of four	
	2.10	51	The correct line has the # symbol removed new_string = new_string + char	3rd in sequence of four	
	2.11	58	The correct line has the # symbol removed new_string = new_string.lower()	4th in sequence of four	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
2 (continued)			Main Program	Do not award marks in this section, where the order of the lines matches the original student code	15
	2.12	68	Input, while, input loop repeats until empty string entered <pre> input_string = input("Enter a word or string: ") while (input_string != ""): . . . input_string = input("Enter a word or string: ") </pre>	- While immediately follows input AND input is last instruction INSIDE while loop - Ignore intervening lines - Ignore over indentation	
	2.13	70 74	String must be prepared before checking to see if it is a palindrome <pre> the_string = prepare_string(input_string) . . . if (is_palindrome(the_string)): </pre>	- Strict order only - Ignore intervening lines - Ignore indentation errors	
			Overall		
	2.14		The solution: - is a reasonable attempt to solve the problem - must translate without syntax errors	- For this mark, some amendments to the code must have been attempted - Indentation errors are syntax errors - Ignore functionality	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
2 (continued)	2.15		Fully functional	Works with test data given in the question paper	15

Input	Output
civic	Original: civic Prepared: civic is a palindrome
CiVic	Original: CiVic Prepared: civic is a palindrome
No melon, no lemon	Original: No melon, no lemon Prepared: nomelonnolemon is a palindrome
Coffee	Original: Coffee Prepared: coffee is not a palindrome
34\$543	Original: 34\$543 Prepared: 34543 is a palindrome
<empty>	Program terminates

```

1 # -----
2 # Constants
3 # -----
4 SPACE = " "
5
6 # -----
7 # Global variables
8 # -----
9 input_string = ""
10 the_string = ""
11
12 # -----
13 # Subprograms
14 # -----
15
16 # =====> Rearrange these lines
17 def is_palindrome(p_string):
18     length = len(p_string)
19     if (length < 2):
20         return (True) # Base case
21     elif (p_string[0] == p_string[length - 1]):
22         return (is_palindrome(p_string[1:length - 1])) # Next two from each end match
23     else:
24         return (False) # Keep moving along
25 # End rearranging
26
27 # =====> Choose the correct line for the subprogram definition
28 #def prepare_string(new_string):
29 def prepare_string(p_string):
30 #def prepare_string(the_string):
31 #def prepare_string(input_string):
32
33     new_string = ""
34
35     # =====> Choose the correct line for the loop
36     for index in range(len(p_string)):
37     #for index in range(1, len(p_string)):
38     #for index in range(1, len(p_string) - 1):
39     #for index in range(len(p_string) - 1):
40         char = p_string[index]
41
42     # =====> Choose the correct line to deal with spaces and punctuation
43     #if (char.isspace()):
44     #if (char.isalpha()):
45     #if (char.strip(SPACE)):
46     if (char.isalnum()):
47
48         # =====> Choose the correct line to create the
49         #new_string = char + [new_string]
50         #new_string = str(char), new_string
51         new_string = new_string + char
52         #new_string = [char, new_string]
53
54     # =====> Choose the correct line to produce the required case
55     #new_string = new_string.islower()
56     #new_string = new_string.isupper()
57     #new_string = not new_string.upper()
58     new_string = new_string.lower()
59
60     return (new_string)
61 # End choosing the correct line
62
63 # -----
64 # Main program
65 # -----
66
67 # =====> Rearrange these lines
68 input_string = input("Enter a string: ")
69 while (input_string != ""): # Loop until empty string inputted
70     the_string = prepare_string(input_string)
71     if (the_string == ""): # Validate input
72         print("Invalid string. Must be only letters and digits. Must not be blank")
73     else: # Process and report
74         if (is_palindrome(the_string)):
75             print("Original: " + input_string + " Prepared: " + the_string + " is a palindrome")
76         else:
77             print("Original: " + input_string + " Prepared: " + the_string + " is not a palindrome")
78     input_string = input("Enter a string: ")
79 # End rearranging
80

```

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
3			Anywhere within the code	• Double quotes and single quotes are interchangeable, as long as they are paired accurately • Ignore omission of ; • Ignore lowercase keywords • Ignore quotation marks, except where noted	15
	3.1	40, 42 49, 51 68, 72 75, 78, 80	The line to execute a query is completed correctly <pre>db_cur.execute(db_query)</pre> The line to retrieve query results is completed correctly <pre>db_result = db_cur.fetchall()</pre>	• Both lines required • Award even if query is incorrect • Award only one instance • Allow fetchone() for fetchall() for this mark	
	3.2	65 85	The line to call the show_result is added <pre>show_result(db_result)</pre>	• Award even if query fails • Award one instance only	
	3.3	46 55	An output message that is fit for purpose is displayed <pre>print("Product 9 has sold " + str(db_result[0][0]) + " times.")</pre>	Only one instance required for this mark Meaningful for the user and informative	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
3 (continued)			Number of different products		15
	3.4	40	The line is completed to identify the keywords for a query and specify a table "SELECT COUNT (*) FROM tbl_products;"	Award red only	
	3.5	40	The line is completed to find the total number "SELECT COUNT (*) FROM tbl_products;"	- Award red only - Award field name for*	
			Number of Product ID 9 sold		
	3.6	49	The line is completed to find the sum "SELECT SUM (quantity) FROM tbl_order_products WHERE product_ID = 9;"	Award red only	
	3.7	49	The line is completed to find the number of product number 9 sold "SELECT SUM (quantity) FROM tbl_order_products WHERE product_ID = 9;"	Award red only	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
3 (continued)			Customers in postcodes 608087%		15
	3.8	58	The line is completed to find all the customers with a similar postcode <pre>"SELECT * FROM tbl_customers WHERE postcode LIKE '608087%';"</pre>	Award red only	
	3.9	58	The line is completed to set the postcode pattern to match <pre>"SELECT * FROM tbl_customers WHERE postcode LIKE '608087%';"</pre>	- Award red only - Clause must be accurate with single or double quotes around postcode and with single % symbol at the end	
			Add the employee table		
	3.10	68	The line is completed to create the employee table <pre>'CREATE TABLE "tbl_employees" ("employee_ID" INTEGER NOT NULL, "address_1" TEXT, "address_2" TEXT, "postcode" TEXT, PRIMARY KEY("employee_ID"))';</pre>	Award red only	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
3 (continued)	3.11	68	The line is completed to set the primary key for the employee table <pre>'CREATE TABLE "tbl_employees" ("employee_ID" INTEGER NOT NULL, "address_1" TEXT, "address_2" TEXT, "postcode" TEXT, PRIMARY KEY("employee_ID"));</pre>	Award red only	15
	Who are the wood workers?				
	3.12	75	The line is completed with the keywords, fields, and table name required for the query <pre>"SELECT trading_name FROM tbl_makers"</pre>	Award red only	
	3.13	76 77	The inner join is completed with the two table names for each side of the join <pre>"INNER JOIN tbl_crafts ON tbl_makers.maker_id = tbl_crafts.maker_id" "INNER JOIN tbl_specialisms ON tbl_specialisms.specialism_id = tbl_crafts.specialism_id"</pre>	- Award red only - Same field on two tables - Same field on two tables - Table names must be supplied or the field names will conflict	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
3 (continued)	3.14	78	The line is completed to specify the condition for the query "WHERE tbl_specialisms.description = 'Woodworking';"	- Award red only - Must have double or single quote around 'Woodworking'	15
	Functionality and refinement				
	3.15		The outputs are accurate	Ignore formatting	

Test Data

Question	Query	Expected output
How many different products are there?	"SELECT COUNT (*) FROM tbl_products;"	There are 12 different products.
How many of product_ID 9 have been sold?	"SELECT SUM (quantity) FROM tbl_order_products WHERE product_ID = 9;"	Product 9 has sold 5 times.
Who are all the customers in postcodes that start with: 608087	"SELECT * FROM tbl_customers WHERE postcode LIKE '608087%';"	The customers who have postcodes that start with 608087 are: (327675, 'Evans', 'Erin', '6080873745') (534980, 'Adams', 'Aspen', '6080878902') (571526, 'Franks', 'Faye', '6080872932')

Question	Query	Expected output
Add a table for the employees	<pre>'CREATE TABLE "tbl_employees" ("employee_ID" INTEGER NOT NULL, ' + \ '"address_1" TEXT, "address_2" TEXT, "postcode" TEXT, ' + \ 'PRIMARY KEY("employee_ID"))';</pre>	No output to the screen
Who are the woodworkers?	<pre>"SELECT trading_name FROM tbl_makers " + \ "INNER JOIN tbl_crafts ON tbl_makers.maker_id = tbl_crafts.maker_id " + \ "INNER JOIN tbl_specialisms ON tbl_specialisms.specialism_id = tbl_crafts.specialism_id " + \ "WHERE tbl_specialisms.description = 'Woodworking';"</pre>	The trading names of the makers who specialise in Woodworking are: (Natural Materials,) (Delicate Designs,)
To see if new table is added, examiner copies this code to the bottom of the file and execute.	<pre>db_query = 'SELECT name FROM sqlite_master WHERE type="table"' db_cur.execute(db_query) db_result = db_cur.fetchall() show_result(db_result) db_query = 'PRAGMA table_info ("tbl_employees")' db_cur.execute(db_query) db_result = db_cur.fetchall() show_result(db_result)</pre>	<pre>('tbl_specialisms',) ('tbl_makers',) ('tbl_customers',) ('tbl_order_products',) ('tbl_orders',) ('tbl_products',) ('tbl_crafts',) ('tbl_employees',) (0, 'employee_ID', 'INTEGER', 1, None, 1) (1, 'address_1', 'TEXT', 0, None, 0) (2, 'address_2', 'TEXT', 0, None, 0) (3, 'postcode', 'TEXT', 0, None, 0)</pre>


```

C:\Users\Cynthia\Desktop\IAL Computer Science\Papers\SAM\U2\16 Consolidation after typesetting & scrutiny\Mark_Scheme_Answers\Q03_Finished.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Q03_Finished.py
1 # -----
2 # Import libraries
3 # -----
4 import sqlite3
5
6 # -----
7 # Constants
8 # -----
9
10 # -----
11 # Global variables
12 # -----
13 db_con = None          # The database connection
14 db_cur = None          # The database cursor
15 db_query = ""          # The SQL query
16 db_result = None       # The result of the query execution
17
18 # -----
19 # Subprograms
20 # -----
21
22 # -----
23 # show_result
24 #   p_list - a list of items
25 #   Displays each item in the list on a separate line
26 # -----
27 def show_result(p_list):
28     for item in p_list:
29         print(item)
30
31 # -----
32 # Main program
33 # -----
34 # Connect to the database. Note, error handling is left out on purpose
35 db_con = sqlite3.connect("CustomCrafts.db")
36 db_cur = db_con.cursor()
37 db_cur.execute("DROP TABLE IF EXISTS tbl_employees;")
38
39 # =====> Complete the SQL to determine how many different products are held in the database.
40 db_query = "SELECT COUNT (*) FROM tbl_products;"
41 # =====> Complete the line to execute the query.
42 db_cur.execute(db_query)
43 # =====> Complete the line to fetch the query results.
44 db_result = db_cur.fetchall()
45 # =====> Add a line to display an appropriate message to the user
46 print("There are " + str(db_result[0][0]) + " different products.")
47
48 # =====> Complete the SQL to determine quantity of product ID 9 that has been sold.
49 db_query = "SELECT SUM (quantity) FROM tbl_order_products WHERE product_ID = 9;"

```

```

50 # =====> Complete the line to execute the query.
51 db_cur.execute(db_query)
52 # =====> Complete the line to fetch the query results.
53 db_result = db_cur.fetchall()
54 # =====> Add a line to display an appropriate message to the user.
55 print("Product 9 has sold " + str(db_result[0][0]) + " times.")
56
57 # =====> Complete the SQL to determine all the customers who have postcodes that start with: 60808
58 db_query = "SELECT * FROM tbl_customers WHERE postcode LIKE '60808%';"
59 # =====> Complete the line to execute the query.
60 db_cur.execute(db_query)
61 # =====> Complete the line to fetch the query results.
62 db_result = db_cur.fetchall()
63 print ("The customers who have postcodes that start with 608087 are: ")
64 # =====> Add a line to call the show_result subprogram to display the results.
65 show_result(db_result)
66
67 # =====> Complete the SQL to add the employee table to the database
68 db_query = 'CREATE TABLE "tbl_employees" ("employee_ID" INTEGER NOT NULL, ' + \
69 ' "address_1" TEXT, "address_2" TEXT, "postcode" TEXT, ' + \
70 ' PRIMARY KEY("employee_ID"));'
71 # =====> Complete the line to execute the query.
72 db_cur.execute(db_query)
73
74 # =====> Complete the SQL to determine the trading names of the makers who specialise in Woodworki
75 db_query = "SELECT trading_name FROM tbl_makers " + \
76 " INNER JOIN tbl_crafts ON tbl_makers.maker_id = tbl_crafts.maker_id " + \
77 " INNER JOIN tbl_specialisms ON tbl_specialisms.specialism_id = tbl_crafts.specialism_id " + \
78 " WHERE tbl_specialisms.description = 'Woodworking';"
79 # =====> Complete the line to execute the query.
80 db_cur.execute(db_query)
81 # =====> Complete the line to fetch the query results.
82 db_result = db_cur.fetchall()
83 print ("The trading names of the makers who specialise in Woodworking are: ")
84 # =====> Add a line to call the show_result subprogram to display the results.
85 show_result(db_result)
86
87 # -----
88 # For examiners only to display all the table names to check if
89 # the new employee table has been added.
90 print("=*80)
91 db_query = 'SELECT name FROM sqlite_master WHERE type="table"'
92 db_cur.execute(db_query)
93 db_result = db_cur.fetchall()
94 show_result(db_result)
95 db_query = 'PRAGMA table_info ("tbl_employees")'
96 db_cur.execute(db_query)
97 db_result = db_cur.fetchall()
98 show_result(db_result)

```

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4			Libraries		18
	4.1	5	Random library is imported <code>import random</code>	Allow any method	
			insertion_sort Subprogram	No return value required, as this subprogram should be a procedure, but ignore if the sorted list is returned. Full marks still available.	
	4.2	26	A for in range() is used for the outer loop <code>for outer_index in range()</code> :		
	4.3	26	The range() function is accurately bound by 1 and the length of the list parameter <code>range(1, len(p_list))</code>		
	4.4	29	First item is acknowledged as already sorted, by setting the target to the item at index 1 <code>target = p_list[outer_index]</code>		

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4 (continued)	4.5	35	The first comparator is set to the second item <code>inner_index = outer_index - 1</code>	See additional guidance below	18
	4.6	43	A while loop is used to implement the inner loop <code>while ((inner_index >= 0) and (p_list[inner_index] > target)):</code>	Award red only	
	4.7	43	The tests for the while loop are both accurate and joined with a Boolean operator <code>while ((inner_index >= 0) and (p_list[inner_index] > target)):</code>	- Award red only - See additional guidance below - Condition 1: <code>inner_index >= 0</code> - Condition 2: <code>p_list[inner_index] > item</code>	
	4.8	47	The value which is greater than the target is shifted to the right <code>p_list[inner_index + 1] = p_list[inner_index]</code>	See additional guidance below	
	4.9	50	The inner_index is decremented <code>inner_index = inner_index - 1</code>		
	4.10	55	The item is moved to the correct position <code>p_list[inner_index + 1] = target</code>	See additional guidance below	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4 (continued)			display_list_matrix Subprogram		18
	4.11	60	A loop is used to process every item in the inputted list <code>for index in range(len(p_list)):</code>	- Allow any method - Use of for <index> in range anticipated - Allow use of for each, with additional counter for column	
	4.12	64	A mechanism for determining when the MAX_WIDTH position has been reached is implemented <code>if (((index + 1) % MAX_WIDTH) == 0):</code>	- Allow any method - Use of % is expected - Use of MAX_WIDTH required	
	4.13	65 67	A mechanism for adding a space, when not at MAX_WIDTH position, and adding a LF, when at MAX_WIDTH position, is implemented <code>print(p_list[index])</code> <code>print(str(p_list[index]) + " ", end = ' ')</code>	Allow any method	
	4.14		Outputs for the number of rows, with MAX_WIDTH number of items each are displayed	Do not award if displaying full list notation, e.g. with commas and square brackets	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4 (continued)			Main program		18
	4.15	76 77 78	Main program: • prints little_list • calls insertion sort subprogram with little_list as an argument • prints the sorted little_list	• No formatting required for little list, so can be in full list notation • Award even if insertion sort does not fully work	
	4.16	82	One hundred random integers between zero and 100 are generated and appended to big_list for count in range(100): bigList.append (random.randint(0, 100))	• Allow any method for loop	
	4.17	88	Main program: • calls display_list_matrix() with big_list as an argument • calls insertion_sort(), with big_list as an argument • calls display_list_matrix(), with big_list as an argument display_list_matrix(big_list) insertion_sort(big_list) display_list_matrix(big_list)	• Award for sequence of accurate calls • Award even if the display list matrix subprogram does not fully work • Award even if the insertion_sort subprogram does not fully work	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
4 (continued)			Functionality		18
	4.18		The outputs are accurate for both little_list and big_list	- Must have called insertion sort with little_list and big_list as a parameter - Do not award for sort method other than the user-devised insertion sort	

Mark point	Guidance
MP 4.5 MP 4.7 MP 4.8 MP 4.10	The inner loop and the location of the insertion can be done by tracking the location of the vacancy on each shift. Either solution is awardable. <pre> for i in range(1,len(pList)): key = pList[i] curPos = i while key < pList[curPos-1] and curPos > 0: pList[curPos] = pList[curPos-1] curPos -= 1 pList[curPos] = key </pre>

```

1  # -----
2  # Import libraries
3  # -----
4  # =====> Import the required library
5  import random
6
7  # -----
8  # Constants
9  # -----
10 MAX_WIDTH = 20
11
12 # -----
13 # Global variables
14 # -----
15 little_list = [44, 22, 11, 33, 55]
16 big_list = []
17
18 # -----
19 # Subprograms
20 # -----
21 def insertion_sort(p_list):
22
23     # =====> Create an outer FOR loop starting at the index position
24     #           of the second item in the list that goes right in the
25     #           list, one item at a time, until there are no more items
26     for outer_index in range(1, len(p_list)):
27
28         # =====> Create a variable that holds the target value
29         target = p_list[outer_index]
30
31         # =====> Create a variable to hold the index value for
32         #           the inner loop that is one to the left of
33         #           the outer index, because it moves along the
34         #           sorted list going left
35         inner_index = outer_index - 1
36
37         # =====> Create an inner WHILE loop that handles shifting.
38         #           The WHILE loop repeats as long as
39         #           the inner index is greater than or equal to 0
40         #           AND
41         #           the value in the list at the inner index
42         #           is greater than the target value
43         while ((inner_index >= 0) and (p_list[inner_index] > target)):
44
45             # =====> Shift the value at the inner index to the right
46             #           by one position
47             p_list[inner_index + 1] = p_list[inner_index]
48
49             # =====> Adjust the inner index one to the left
50             inner_index = inner_index - 1
51

```

```

52         # =====> A location is identified. Insert the target value
53         #             into the gap that is one to the right
54         #             of the inner index
55         p_list[inner_index + 1] = target
56
57
58     def display_list_matrix(p_list):
59         # =====> Set up a loop to process every item in the inputted list
60         for index in range(len(p_list)):
61             # =====> Check if the item is at the MAX_WIDTH location
62             #             If it is, then print with the line feed
63             #             Otherwise, print the item followed by a SPACE
64             if ((index + 1) % MAX_WIDTH) == 0:
65                 print(p_list[index])
66             else:
67                 print(str(p_list[index]) + " ", end = '')
68
69     # -----
70     # Main program
71     # -----
72
73     # =====> Display the unsorted little list (no formatting required),
74     #             Call the insertion sort subprogram to sort the little list
75     #             Display the sorted little list (no formatting required)
76     print(little_list)
77     insertion_sort(little_list)
78     print(little_list)
79     print("\n")
80
81     # =====> Fill the big list with 100 random integers between 0 and 100
82     for count in range(100):
83         big_list.append(random.randint(0, 100))
84
85     # =====> Display the unsorted big list in rows of MAX_WIDTH items
86     #             Call the insertion_sort to sort the big list
87     #             Display the sorted big list in rows of MAX_WIDTH items
88     display_list_matrix(big_list)
89     print("\n")
90     insertion_sort(big_list)
91     display_list_matrix(big_list)
92

```

```

93  # -----
94  # Examiner testing only
95  # -----
96
97  # Copy this line to immediately follow the creation of the big_list of
    random numbers
98  # It makes a copy that can be used to double check that the code is
    working
99  unsortedBigList = big_list.copy()
100
101  print("\n")
102  print("-"*70)
103  sortedBigList = sorted(unsortedBigList)
104  # sortedBigList[0] = 888          # Show a real error, just for debug
105  mismatches = [item for item in big_list if item not in sortedBigList] + \
106               [item for item in sortedBigList if item not in big_list]
107  if (len(mismatches) != 0):
108      print("Error - Two big lists do not match.  Mismatches: ")
109      print(mismatches)
110  else:
111      print("Pass - Two big lists match")
112

```

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5			Libraries		20
	5.1	5	Import datetime library <code>import datetime</code>	Allow any method	
			Main loop		
	5.2	117	A loop is used to process every row in the data, bounded by the length of the array <code>while (index < len(data)):</code>	- Allow any method - Loop to be bound on length of data structure	
	5.3	119 120	Two-dimensional indexing is used to access the data <code>record = data[index]</code> <code>sensor_ID = record[ID]</code>	Award for/while for first dimension with indexing for second dimension	
			Identifying invalid readings		
	5.4	76	A range check is used to determine if a reading is valid <code>p_record[READING] < MIN_READING</code> or <code>(p_record[READING] > MAX_READING)</code>	- Allow reverse relational operator if the test is for invalid - Allow literals instead of constants	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)	5.5	125 126 127	A data structure is used to track bad records <code>if sensor_ID not in bad_sensor_IDs: bad_sensor_IDs.add(sensor_ID) bad_sensor_records.append(record)</code>	• 1D or 2D list anticipated • Only for bad records • Ignore tracking bad IDs	20
	A user-defined subprogram				
	5.6	73 81	One subprogram definition is constructed accurately with parameters <code>def is_valid_reading(p_record):</code>	• Likely choices for subprograms are validation and reporting <code>def report_faulty_sensors(p_data):</code>	
	5.7	123 131	One user-devised subprogram is called and the calling code passes in an argument <code>if (not is_valid_reading(record)):</code>	• Likely location for these calls is in the main program <code>report_faulty_sensors(bad_sensor_records)</code>	
	Output (display and file)				
	5.8	82	The correct file is opened for writing only and closed, as required <code>out_file = open(OUT_FILE, "w")</code>	• Allow literal file name • Allow 'with open' if used accurately	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)	5.9	107	More than one line containing relevant information, is written to the file <pre> for index in range(len(p_data)): out_file.write(f"{record[ID]:^5} {record[LOCATION]:^22} {my_date_time: %Y-%m-%d %H:%M:%S} {record[READING]:>7.1f}\n") </pre>	• Allow even if information is not accurate or is inaccurately formatted • Do not award for headers and separators only	20
	5.10	105	Invalid readings are displayed in the required format <pre> print(f"{record[ID]:^5} {record[LOCATION]:^22} {my_date_time:%Y-%m-%d %H:%M:%S} {record[READING]:>7.1f}") </pre>	• Headings and separator displayed • Centre alignment for integers and text • Right alignment for real numbers • Allow different column widths	
	5.11	103	Functions from the datetime library are used to convert timestamp to a human-readable format <pre> record = p_data[index] my_date_time = datetime.datetime.fromtimestamp(record[TIMESTAMP]) </pre>		

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
5 (continued)			Design		20
	5.12		Tabular output meets alignment requirements and is fit for purpose		
	5.13		A single pass through the data is implemented		
	5.14		Reduction of side effects is demonstrated by original data structure remaining unchanged for the duration of the program	For award of this mark, there must be code that attempts to access the dataset	
			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 item in the dataset is processed - Invalid readings are identified - 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: - Sensors with invalid readings are tracked - Invalid sensors are reported to the display	

Question number	Mark Point	Appx. Line	Answer Award marks as shown	Additional Guidance	Mark
	5.20		The outputs are accurate	• The displayed outputs match requirements • The outputs stored in the file match requirements • Date/Time UTC displayed and stored in the file in correct format	

Output

There are 4 faulty sensors			
ID	Location	Date/Time	Reading
16	Hyde Park Street	2024-11-05 02:50:00	-33.7
18	Abbey Wood	2024-11-01 00:10:00	55.7
47	Greenwich Park	2024-11-05 06:30:00	0.0
125	Green Park	2024-11-03 04:50:00	122.8

```

1  # -----
2  # Import libraries
3  # -----
4  # =====> Write your code here
5  import datetime
6
7  # -----
8  # Constants
9  # -----
10 OUT_FILE = "Q05_Out.txt"
11 LF = "\n"
12
13 MIN_READING = 10.0          # Minimum valid reading
14 MAX_READING = 51.9          # Maximum valid reading
15
16 ID = 0                      # Field indexes in data record
17 LATITUDE = 1
18 LONGITUDE = 2
19 LOCATION = 3
20 TIMESTAMP = 4
21 READING = 5
22
23 # -----
24 # Global variables
25 # -----
26 # sensor id, latitude, longitude, location, timestamp, reading
27 data = [[16, 51.51273, -0.16709, "Hyde Park Street", 1730775000, -33.7],
28         [16, 51.51273, -0.16709, "Hyde Park Street", 1730775600, 19.1],
29         [16, 51.51273, -0.16709, "Hyde Park Street", 1730776200, 31.5],
30         [16, 51.51273, -0.16709, "Hyde Park Street", 1730776800, 29.4],
31         [16, 51.51273, -0.16709, "Hyde Park Street", 1730777400, 13.9],
32         [18, 51.49087, 0.12227, "Abbey Wood", 1730419200, 23.8],
33         [18, 51.49087, 0.12227, "Abbey Wood", 1730419800, 55.7],
34         [18, 51.49087, 0.12227, "Abbey Wood", 1730420400, -2.5],
35         [18, 51.49087, 0.12227, "Abbey Wood", 1730992800, 22.2],
36         [18, 51.49087, 0.12227, "Abbey Wood", 1730993400, 42.9],
37         [47, 51.47229, 0.00359, "Greenwich Park", 1730419200, 51.9],
38         [47, 51.47229, 0.00359, "Greenwich Park", 1730419800, 19.8],
39         [47, 51.47229, 0.00359, "Greenwich Park", 1730420400, 49],
40         [47, 51.47229, 0.00359, "Greenwich Park", 1730788200, 0],
41         [47, 51.47229, 0.00359, "Greenwich Park", 1730788800, 96.5],
42         [54, 51.524, -0.14714, "Regent's Park", 1730492400, 39.9],
43         [54, 51.524, -0.14714, "Regent's Park", 1730493000, 29.7],
44         [54, 51.524, -0.14714, "Regent's Park", 1730493600, 40],
45         [54, 51.524, -0.14714, "Regent's Park", 1730494200, 46.7],
46         [61, 51.50972, -0.0758, "Tower of London", 1730647200, 11.8],
47         [61, 51.50972, -0.0758, "Tower of London", 1730647800, 25],
48         [61, 51.50972, -0.0758, "Tower of London", 1730648400, 41.7],
49         [61, 51.50972, -0.0758, "Tower of London", 1730649000, 36.7],
50         [61, 51.50972, -0.0758, "Tower of London", 1730649600, 26.3],
51         [125, 51.50406, -0.14733, "Green Park", 1730609400, 122.8],
52         [125, 51.50406, -0.14733, "Green Park", 1730610000, 0],
53         [125, 51.50406, -0.14733, "Green Park", 1730610600, 12.1],
54         [125, 51.50406, -0.14733, "Green Park", 1730611200, 40.4],
55         [144, 51.51077, 0.03525, "London City Airport", 1730517600, 37],
56         [144, 51.51077, 0.03525, "London City Airport", 1730518200, 34],
57         [144, 51.51077, 0.03525, "London City Airport", 1730518800, 30.1],
58         [144, 51.51077, 0.03525, "London City Airport", 1730519400, 22.7],
59         [144, 51.51077, 0.03525, "London City Airport", 1730520000, 49.2]]
60

```

```

61 # =====> Write your code here
62 bad_sensor_IDs = set()           # Just tracks sensors with bad readings
63 bad_sensor_records = []         # For the whole first record of a bad sensor
64
65 # General use
66 index = 0
67
68 # -----
69 # Subprograms
70 # -----
71 # =====> Write your code here
72
73 def is_valid_reading(p_record):
74     valid = True                 # Assume good reading
75
76     if ((p_record[READING] < MIN_READING) or (p_record[READING] > MAX_READING)):
77         valid = False
78
79     return (valid)
80
81 def report_faulty_sensors(p_data):
82     out_file = open(OUT_FILE, "w")
83
84     # Report the number of bad sensors found to display and to file
85     print(LF)
86     print("There are " + str(len(bad_sensor_IDs)) + " faulty sensors")
87
88     out_file.write("There are " + str(len(bad_sensor_IDs)) + " faulty sensors\n")
89
90     # Report the header information to display and to file
91     print(LF)
92     print(f"{'ID':<5} {'Location':^22} {'Date/Time':^20} {'Reading':^7}")
93     print("-" * 60)
94
95     out_file.write(LF)
96     out_file.write(f"{'ID':<5} {'Location':^22} {'Date/Time':^20} {'Reading':^7}\n")
97     out_file.write("-" * 60)
98     out_file.write(LF)
99
100    # Report sensor information for each faulty sensor to display and to file
101    for index in range(len(p_data)):
102        record = p_data[index] # More readable
103        my_date_time = datetime.datetime.fromtimestamp(record[TIMESTAMP])
104
105        print(f"{record[ID]:^5} {record[LOCATION]:^22} {my_date_time:%Y-%m-%d %H:%M:%S} {record[READING]:>7.1f}")
106
107        out_file.write(
108            f"{record[ID]:^5} {record[LOCATION]:^22} {my_date_time:%Y-%m-%d %H:%M:%S} {record[READING]:>7.1f}\n")
109
110    out_file.close()

```

```

111 # -----
112 # Main program
113 # -----
114 # =====> Write your code here
115
116 # Look at all the records in the data
117 while (index < len(data)):
118
119     record = data[index]
120     sensor_ID = record[ID]          # Track to reduce indexing later
121
122     # Range check for out of bounds readings
123     if (not is_valid_reading(record)):
124         # If this bad sensor is not seen before, track it
125         if sensor_ID not in bad_sensor_IDs:
126             bad_sensor_IDs.add(sensor_ID)
127             bad_sensor_records.append(record)
128
129     index = index + 1                # Move to the next record
130
131 report_faulty_sensors(bad_sensor_records)
132

```

Pearson Edexcel International Advanced Level

Sample assessment material

Paper
reference

WCP02/01

Computer Science

**Unit 2: Practical Programming and Problem-solving
Programming Language Subset (PLS)**

Resource Booklet

Do not return this Booklet with the question paper.

S89028A

©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 2 assessment. This does not stop a teacher/student 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 2, 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

<pre>for <id> in <structure>: <command></pre>	Executes <command> for each element of <structure>, in one dimension.
<pre>for <id> in range (<start>, <stop>): <command></pre>	Executes <command> a fixed number of times, based on the numbers generated by the range function. <stop> is required. <start> is optional.
<pre>for <id> in range (<start>, <stop>, <step>): <command></pre>	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

<pre>while <condition>: <command></pre>	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
<code>/</code>	Division
<code>*</code>	Multiplication
<code>**</code>	Exponentiation
<code>+</code>	Addition
<code>-</code>	Subtraction
<code>//</code>	Integer division
<code>%</code>	Modulus

Relational

Relational operator	Description
<code>==</code>	Equal to
<code>!=</code>	Not equal to
<code>></code>	Greater than
<code>>=</code>	Greater than or equal to
<code><</code>	Less than
<code><=</code>	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>del <list> [<index>]</code>	Removes the item at <index> from <list>.
<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>.

Dictionary

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

<code><aDict>.keys ()</code>	Returns all the keys in <code><aDict></code> . Use <code>list ()</code> to convert the returned result to a list.
<code><aDict>.values ()</code>	Returns all the values in <code><aDict></code> . Use <code>list ()</code> 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 <code><item></code> 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>.

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 <string> where the first letter is capitalised.
<code><string>.find (<substring>, <start>, <end>)</code>	Returns the location of the first instance of <substring> in the original <string>, reading from left to right. <start> is the index to begin the find. The default is zero. <end> is the index to stop the find. The default is the end. <string>. Returns -1, if not found.
<code><string>.format (<values>)</code>	Formats <values> and puts them into <string>. The content of <string> is described by symbols and placeholders.
<code><string>.index (<substring>, <start>, <end>)</code>	Returns the location of the first instance of <substring> found in the original <string> as read from left to right. Raises an exception if not found. <substring> is required. <start> and <end> are optional. The default value for <start> is zero. The default value for <end> is the end of <string>.
<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 <string> are lowercase.
<code><string>.isnumeric ()</code>	Returns true if all the characters in <string> are the values 0 to 9.
<code><string>.isspace ()</code>	Returns true if all the characters in <string> are whitespaces.
<code><string>.isupper ()</code>	Returns True, if all characters in <string> are uppercase.
<code>len (<string>)</code>	Returns the length of <string>.
<code><string>.lower ()</code>	Returns original <string> 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 a Coordinated Universal Time (UTC).
Text files	Store as a text string in ISO 8601 format.
SQLite3 library	Store as a text string in ISO 8601 format.

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"

Relational databases

The SQLite3 library module is used to support relational database operations.

Communication

<code><connection>.close ()</code>	Closes the database connection, when no longer required.
<code><cursor>.commit ()</code>	Ensures any outstanding operations are immediately committed against the database <code><cursor></code> .
<code>sqlite3.connect (<database>)</code>	Returns a connector handle to <code><database></code> .
<code><connection>.cursor ()</code>	Returns a cursor to the database <code><connection></code> .
<code><cursor>.execute (<expression>)</code>	Executes <code><expression></code> against the database <code><cursor></code> .
<code><cursor>.fetchall ()</code>	Retrieves all the rows in the result of the last execution call on <code><cursor></code> .
<code><cursor>.fetchone ()</code>	Retrieves the next row in the result of the last execution call on <code><cursor></code> .

Tables

<code>CREATE TABLE <table> (<name1 type constraint>, <name2 type constraint>, . . .);</code>	Creates <code><table></code> with columns of the indicated names to hold values of <code><type></code> . The constraint is optional.
<code>DROP TABLE <table>;</code>	Deletes an existing table. All the table contents are destroyed.

Constraint keywords

DEFAULT	Assigns a default value for a column where no value is supplied.
FOREIGN KEY	Ensures that links between tables are not destroyed.
NOT NULL	Null values not allowed.
PRIMARY KEY	Uniquely identifies each row. Combination of NOT NULL and UNIQUE.
UNIQUE	No duplicates are allowed.

Type keywords

BLOB	The value is a blob of data, stored exactly as it was input.
INTEGER	The value is a signed integer, stored in 0, 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
NULL	The value is the NULL value.
REAL	The value is a floating- point.
TEXT	The value is a text string, stored using the database encoding. Use this type for storing datetimes in ISO 8601 format.

Manipulating records

DELETE FROM <table>;	Deletes all the records in <table>. The table structure is not destroyed.
DELETE FROM <table>; WHERE <criteria>;	Deletes all the records in <table> that match the <criteria>.
INSERT INTO <table> (<columns>) VALUES (<values>;	Inserts a record into <table>. The column names are given by <columns>. The values to put into the <columns> are given by <values>. The ordering must match.
TRUNCATE TABLE <table>;	Deletes all the data in <table>, but does not delete the <table> structure.
UPDATE <table> SET <field1> = <value1>, <field2> = <value2>, . . . WHERE <criteria>	Modifies the existing records in an existing table.

Retrieving data

SELECT <columns> FROM <table> WHERE <criteria>;	Returns a set of records containing <columns> from the existing <table>. The WHERE clause is optional. For all the columns, use *.
SELECT DISTINCT <columns> FROM <table> WHERE <criteria>;	Returns a set of records containing <columns> from the existing <table>. The returned values are unique across columns. The WHERE clause is optional.

Filtering and refining data

WHERE <criteria>	Returns only those records that meet <criteria>.
<expr1> AND <expr2>	Includes only those records where both <expr1> and <expr2> are true.
BETWEEN <begin> AND <end>	Used to select values in a range. <begin> and <end> values are included.
DISTINCT	Used to ensure there are no duplicate rows in the result.
LIKE	Used to find records that match a pattern. % indicates zero, one, or more characters. _ indicates a single character.
IS NOT NULL	Used to test for non-empty values.
IS NULL	Used to test for empty values.
NOT <condition>	Used to return only records where <condition> is not true.
<expr1> OR <expr2>	Includes only those records where either or both expressions are true.
= <> > >= < <=	Relational operators that may be used in creating <criteria>, <expression>, or <condition>.
<expr1> UNION <expr2>	Returns the combined results from two SELECT statements.

Presentation

ASC	Sorts returned records in ascending order.
DESC	Sorts returned records in descending order.
GROUP BY <column>	Used with aggregate functions to group the returned records.
ORDER BY <column>	Specifies the <column> on which to base the ordering.

Aggregation

HAVING <criteria>	May be used instead of WHERE for aggregate functions. The <criteria> is made up of an aggregate function with a condition.
AVG (<column>)	Returns the mean of <column> where <column> holds numbers.
COUNT (<column>)	Returns the number of rows that match the criteria given in the WHERE clause. Null values will not be counted. Use * for all rows in a table. Null values will be counted.
MAX (<column>)	Returns the largest value in a <column>
MIN (<column>)	Returns the smallest value in a <column>
SUM (<column>)	Returns the sum of <column> where <column> holds numbers.

SQLITE_LOCKED	6	A table in the database is locked
SQLITE_NOMEM	7	A memory allocation failed
SQLITE_READONLY	8	Attempt to write a readonly database
SQLITE_INTERRUPT	9	Operation terminated by sqlite3_interrupt()
SQLITE_IOERR	10	Some kind of disk I/O error occurred
SQLITE_CORRUPT	11	The database disk image is malformed
SQLITE_NOTFOUND	12	Unknown opcode in sqlite3_file_control()
SQLITE_FULL	13	Insertion failed because database is full
SQLITE_CANTOPEN	14	Unable to open the database file
SQLITE_PROTOCOL	15	Database lock protocol error
SQLITE_EMPTY	16	Internal use only
SQLITE_SCHEMA	17	The database schema changed
SQLITE_TOOBIG	18	String or BLOB exceeds size limit
SQLITE_CONSTRAINT	19	Abort due to constraint violation
SQLITE_MISMATCH	20	Data type mismatch
SQLITE_MISUSE	21	Library used incorrectly
SQLITE_NOLFS	22	Uses OS features not supported on host
SQLITE_AUTH	23	Authorization denied
SQLITE_FORMAT	24	Not used
SQLITE_RANGE	25	2nd parameter to sqlite3_bind out of range
SQLITE_NOTADB	26	File opened that is not a database file
SQLITE_NOTICE	27	Notifications from sqlite3_log()
SQLITE_WARNING	28	Warnings from sqlite3_log()
SQLITE_ROW	100	sqlite3_step() has another row ready
SQLITE_DONE	101	sqlite3_step() has finished executing

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.

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 <a> <= X <= .
<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.

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 $ \text{<x>} $.
<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>.

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.

Turtle Graphics

The default mode for the PLS turtle is “standard”. This means that when a turtle is created, it initially points to the right (east) and angles are counterclockwise. You can change modes using `turtle.mode ()`.

The turtle window is one size and the turtle drawing canvas (inside the window) can be a different size. To make the turtle window bigger, a screen needs to be created and set up.

Here is an example:

```
WIDTH = 800  
HEIGHT = 400  
screen = turtle.Screen ()  
screen.setup (WIDTH, HEIGHT)
```

In some development environments, the turtle window will close as soon as the program completes. There are two ways to keep it open:

Add `turtle.done ()` as the last line in the code file. This will keep the window open until closed with the exit cross in the upper right-hand corner. It also allows scrollbars on the window.

Add a line asking for keyboard input, such as `input ()`, as the last line. This will keep the window open until the user presses a key in the console session. The scrollbars will not work.

Window and drawing canvas

<code><window>.setup (<width>, <height>)</code>	Sets the size of the drawing canvas to <width>, <height> in pixels. Requires use of <code>turtle.Screen ()</code> to create <screen> first.
<code>turtle.done ()</code>	Use as the last line of the file to keep the turtle window open until it is closed using the exit cross in the upper right-hand corner of the window.
<code>turtle.mode (<type>)</code>	<type> is one of the strings <code>standard</code> or <code>logo</code> . A turtle in <code>standard</code> mode, initially points to the right (east) and angles are counter clockwise. A turtle in <code>logo</code> mode, initially points up (north) and angles are clockwise.
<code>turtle.Screen ()</code>	Returns a variable to address the turtle window. Use with <code><window>.setup ()</code> .

Creation, visibility and movement

<code><turtle> = turtle.Turtle ()</code>	Creates a new turtle with the variable name <code><turtle></code> .
<code><turtle>.back (<steps>)</code>	Moves backward (opposite-facing direction) for number of <code><steps></code> .
<code><turtle>.forward (<steps>)</code>	Moves forward (facing direction) for number of <code><steps></code> .
<code><turtle>.hideturtle ()</code>	Makes the <code><turtle></code> invisible.
<code><turtle>.left (<degrees>)</code>	Turns counterclockwise the number of <code><degrees></code> .
<code><turtle>.right (<degrees>)</code>	Turns clockwise the number of <code><degrees></code> .
<code><turtle>.showturtle ()</code>	Makes <code><turtle></code> visible.
<code><turtle>.speed (<value>)</code>	The <code><value></code> can be set to "fastest", "fast", "normal", "slow", "slowest". Alternatively, use the numbers 1 to 10 to increase speed. The value of 0 is the fastest.

Positioning and direction

<code>turtle>.home ()</code>	Moves to canvas origin (0, 0).
<code><turtle>.reset ()</code>	Clears the drawing canvas, sends the turtle home and resets variables to default values.
<code><turtle>.setheading (<degrees>)</code>	Sets the orientation to <code><degrees></code> .
<code><turtle>.setposition (<x>, <y>)</code>	Positions the turtle at coordinates (<code><x></code> , <code><y></code>).

Filling shapes

<code><turtle>.begin_fill ()</code>	Call just before drawing a shape to be filled.
<code><turtle>.end_fill ()</code>	Call just after drawing the shape to be filled. You must call <code><turtle>.begin_fill ()</code> before drawing.
<code><turtle>.fillcolor (<colour>)</code>	Sets the colour used to fill. The input argument is a string.

Controlling the pen

<code><turtle>.pencolor (<colour>)</code>	Sets the colour of the pen. The input argument is a string or an RGB colour.
<code><turtle>.pendown ()</code>	Puts the pen down.
<code><turtle>.pensize (<width>)</code>	Makes the pen the size of <code><width></code> (positive number).
<code><turtle>.penup ()</code>	Lifts the pen up.

Circles

<code><turtle>.circle (<radius>, <extent>)</code>	Draws a circle with the given <code><radius></code> . The centre is the <code><radius></code> number of units to the left of the turtle. That means, the turtle is sitting on the edge of the circle. The parameter <code><extent></code> does not need to be given, but provides a way to draw an arc, if required. An extent of 180 would be half a circle.
---	---

Colour strings

blue	black	green	yellow
orange	red	pink	purple
indigo	olive	lime	navy
orchid	salmon	peru	sienna
white	cyan	silver	gold

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.

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: GQ000051

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