AQA
Realising potential

# GCSE COMPUTER SCIENCE (8525)

## Marked responses for Python

Understand how different levels are achieved and how to interpret the mark scheme.

# EXAMPLE RESPONSES

Version 1.2

## Question 5

Write a Python program that allows a taxi company to calculate how much a taxi fare should be.

The program should:

- allow the user to enter the journey distance in kilometres (no validation is required)
- allow the user to enter the number of passengers (no validation is required)
- calculate the taxi fare by
  - charging £2 for every passenger regardless of the distance
  - charging a further £1.50 for every kilometre regardless of how many passengers there are
- output the final taxi fare.

You **should** use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

**[7 marks]**

## Mark scheme

| Question | Marking guidance |
|---|---|
| 05 | **2 marks for AO3 (design) and 5 marks for AO3 (program)**<br><br>**Program Design**<br><br>**Mark A** for using meaningful variable names throughout (even if logic is incorrect);<br><br>**Mark B** for using suitable data types throughout (distance can be real or integer, passengers must be integer, fare must be real);<br><br>**Program Logic**<br><br>**Mark C** for getting user input for the distance in an appropriate place;<br><br>**Mark D** for getting user input for the number of passengers in an appropriate place;<br><br>**Mark E** for a fare that correctly charges £2 per passenger;<br><br>**Mark F** for a fare that correctly charges £1.50 for every kilometre;<br><br>**Mark G** for outputting the correct final fare;<br><br>**I.** Case of program code<br><br>**Maximum 6 marks** if any errors in code. |

## Response 1

```
fare = 0
distance = int(input("What is the distance? "))
passengers = int(input("How many passengers? "))
```

**Marks awarded: 3**

This response gets **Mark A** as meaningful variable names (`distance`, `fare`, `passengers`) have been used throughout. It does not get **Mark B** as `fare` does not have an appropriate data type – it is not always going to be a whole number so integer was not a suitable choice. The data type for distance and passengers are appropriate but this is not enough to get the mark.

**Marks C** and **D** are awarded as the program gets values for both the distance and passengers from the user. There is no attempt at writing code for **Marks E**, **F** and **G**.

There are no syntax errors in the code.

## Response 2

```
fare = 0.0
distance = int(input())
passengers = int(input())
```

**Marks awarded: 4**

This response gets **Mark A** as meaningful variable names (`distance`, `fare`, `passengers`) have been used throughout. It also gets **Mark B** as all three variables have an appropriate data type.

**Marks C** and **D** are awarded as the program gets values for both the distance and passengers from the user. There are no prompts telling the user what data to enter but these were not specified as being needed in the question and are not required by the mark scheme. There is no attempt at writing code for **Marks E**, **F** and **G**.

There are no syntax errors in the code.

## Response 3

```
distance = int(input())
passengers = int(input()
fare = 2 * passengers
fare = fare + 1.5 * distance
```

**Marks awarded: 6**

This response gets **Mark A** as meaningful variable names (`distance`, `fare`, `passengers`) have been used throughout. It also gets **Mark B** as all three variables have an appropriate data type.

**Marks C** and **D** are awarded as the program gets values for both the distance and passengers from the user. There are no prompts telling the user what data to enter but these were not specified as being needed in the question and are not required by the mark scheme. There is a minor syntax error on the second line as there is a bracket missing at the end of the line but this does not affect the logic flow so **Mark D** can still be awarded.

The code for **Marks E** and **F** is correct but the code for **Mark G** has been omitted.

## Response 4

```
distance = int(input())
passengers = int(input())
fare = 2 * passengers
fare = fare + 1.5 * passengers
prit(fare)
```

**Marks awarded: 6**

This response gets **Mark A** as meaningful variable names (`distance`, `fare`, `passengers`) have been used throughout. It also gets **Mark B** as all three variables have an appropriate data type.

**Marks C** and **D** are awarded as the program gets values for both the distance and passengers from the user. There are no prompts telling the user what data to enter but these were not specified as being needed in the question and are not required by the mark scheme.

The code for **Mark E** is correct but the code for **Mark F** contains a logical error as it multiplies by `passengers` instead of `distance` and is therefore not awarded. **Mark G** is awarded, even though the value displayed will be incorrect due to the logical error on the previous line. The misspelling of `print` does not prevent the awarding of **Mark G** as the meaning is clear.

## Response 5

```
dis = float(input())
pass = int(input())
fare = 2 * pass
fare = fare + 1.5 * dis
print(fare)
```

**Marks awarded: 7**

This response gets **Mark A** as meaningful variable names (`dis`, `fare`, `pass`) have been used throughout. It also gets **Mark B** as all three variables have an appropriate data type.

**Marks C** to **G** are all awarded.

# Question 6

Write a Python program that inputs a password and checks if it is correct.

Your program should work as follows:

- input a password and store it in a suitable variable
- if the password entered is equal to `secret` display the message `Welcome`
- if the password entered is not equal to `secret` display the message `Not welcome.`

You **should** use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

**[5 marks]**

## Mark scheme

| Question | Marking guidance |
|---|---|
| 06 | **2 marks for AO3 (design), 3 marks for AO3 (program)**<br><br>**Program Design**<br><br>**Mark A** for the use of a selection construct (even if the logic is incorrect);<br><br>**Mark B** for the correct, consistent use of meaningful variable names throughout (even if the code would not work);<br><br>**Program Logic**<br><br>**Mark C** for using user input and storing the result in a variable correctly;<br><br>**Mark D** for a correct expression that checks if the entered password is `'secret'` (even if the syntax is incorrect);<br><br>**Mark E** for outputting `Welcome` and `Not welcome` correctly in logically separate places such as the `IF` and `ELSE` part of selection;<br><br>**I.** Case of output strings for **Mark E**, but spelling must be correct.<br><br>**I.** Case of program code<br><br>**Maximum 4 marks** if any errors in code. |

## Response 1

```
password = input()
```

**Marks awarded: 2**

This response gets **Mark B** as a meaningful variable name has been used. It also gets **Mark C** as, although no variable declaration has been shown, data is obtained from the user and stored in a variable. No attempt at code for the other three marks.

While this is not a very good answer to the question it does show good exam technique – the student has written the (small amount of) code they knew how to do and been able to obtain some marks as a result of doing so.

## Response 2

```
Secret = ""
password = input()
if password == secret:
  print("Welcome")
else
  print("Not welcome")
```

**Marks awarded: 4**

This response gets **Mark B** as a meaningful variable name has been used. There is an additional variable declared that is not needed, but this is ignored when marking.

It also gets **Mark C** as data is obtained from the user and stored in a variable. There is an attempt at a selection structure so **Mark A** is awarded but not **Mark D** as the condition is incorrect (comparing to a variable instead of the string "`secret`"). This answer also gets **Mark E**.

There is a minor syntax error in the last line as a colon is missing after the `else`. This syntax error does not prevent the awarding of any marks as it does not affect the logic flow.

## Response 3

```
password = input()
if password = "secret":
  print("Correct password")
else:
  print("Incorrect password")
```

**Marks awarded: 4**

This response gets **Mark B** as a meaningful variable name has been used.

It also gets **Mark C** as data is obtained from the user and stored in a variable. There is an attempt at a selection structure so **Mark A** is awarded, as is **Mark D** as the condition is logically correct. This answer does not get **Mark E** though as alternative messages to those specified in the question have been used.

There is a minor syntax error as = has been used instead of == in the condition. This syntax error does not prevent the awarding of any marks as it does not affect the logic flow.

## Response 4

```
Answer = "secret"
password = input()
if password != answer:
  print("Not welcome")
else:
  print("Welcome")
```

**Marks awarded: 5**

This response gets full marks.

An additional variable, called `Answer`, has been created, this was not needed but it has been used to store the string "secret" and then used appropriately in the condition for the selection structure (the case is ignored).

The condition in the selection structure is not the most obvious one to use but, as the order of the two outputs is correct for this condition, it is logically equivalent to the fully correct answer shown on the specimen mark scheme.

## Response 5

```
password = input()
if (password == "secret"):
  print("Welcome")
else:
  print("Not welcome")
```

**Marks awarded: 5**

This response gets full marks.

The brackets around the condition in the selection structure are not needed but do not change the functionality of the program code.

## Question 8

Write a Python program that inputs a character and checks to see if it is lowercase or not.

Your program should work as follows:

- gets the user to enter a character and store it in a suitable variable
- determines if the entered character is a lowercase character
- outputs `LOWER` if the user has entered a lowercase character
- outputs `NOT LOWER` if the user has entered any other character.

You **should** use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

**[7 marks]**

## Mark scheme

| Question | Marking guidance |
|---|---|
| 08 | **3 marks for AO3 (design), 4 marks for AO3 (program)**<br><br>**Program Design**<br><br>**Mark A** for the idea of inputting a character and checking if it is lower case (even if the code would not work);<br><br>**Mark B** for the use of a selection construct (even if the logic is incorrect);<br><br>**Mark C** for the correct, consistent use of meaningful variable names throughout (even if the code would not work);<br><br>**Program Logic**<br><br>**Mark D** for using user input correctly;<br><br>**Mark E** for storing the result of user input in a variable correctly;<br><br>**Mark F** for a correct expression/method that checks if the character is lowercase;<br><br>**Mark G** for outputting `LOWER` and `NOT LOWER` correctly in logically separate places such as the `IF` and `ELSE` part of selection;<br><br>**I.** Case of output strings for **Mark G**, but spelling must be correct.<br><br>**I.** Case of program code<br><br>**Maximum 6 marks** if any errors in code. |

## Response 1

```
character = input()
if
  print("LOWER")
else
  print("NOT LOWER")
```

**Marks awarded: 4**

This response gets **Mark C** as a meaningful variable name has been used and **Mark E** as data is obtained from the user and stored in a variable. There is an attempt at a selection structure so **Mark B** is awarded and also **Mark G** as the two messages are in logically sensible places. The omission of the colon does not affect the logic and so is ignored.

There is no attempt at checking if the character is lower case or not (so does not get **Mark F**) and the user's input is not used in the program (so does not get **Mark D**). There is insufficient evidence for **Mark A** as there is no attempt at checking if the character is lower case or not.

This response shows a good approach when answering an exam question with a difficult condition to get correct. User input has been obtained and a selection structure put in the correct place in the code with the appropriate code put in the `if` and `else` parts of the structure.

## Response 2

```
character = input()
if character is lower:
  print("LOWER")
else:
  print("NOT LOWER")
```

**Marks awarded: 6**

This response gets **Mark C** as a meaningful variable name has been used and **Mark E** as data is obtained from the user and stored in a variable. There is an attempt at a selection structure so **Mark B** is awarded and also **Mark G** as the two messages are in logically sensible places.

There is an attempt at checking if the character is lower case or not but it does not get **Mark F** as this is not close enough to the correct syntax. The user's input is used in the attempt at the condition in the selection structure (so it does get **Mark D**). There is sufficient evidence for **Mark A** as there is a clear attempt at checking if the character is lower case or not.

This response shows a good approach when answering an exam question with a difficult condition to get correct. User input has been obtained and a selection structure put in the correct place in the code with the appropriate code put in the `if` and `else` parts of the structure. An attempt has been made at getting the condition in the selection structure correct which has enabled the student to get more marks than if they had not attempted the condition at all.

## Response 3

```
character = input()
if character > = "a" or character < = "z":
  print("LOWER")
else:
  print("NOT LOWER")
```

**Marks awarded: 6**

This response gets **Mark C** as a meaningful variable name has been used and **Mark E** as data is obtained from the user and stored in a variable.

There is an attempt at a selection structure so **Mark B** is awarded and also **Mark G** as the two messages are in logically sensible places.

There is an attempt at checking if the character is lower case or not so it gets **Mark A** but it does not get **Mark F** as while the syntax is correct the logic of the program is not (should be `and` instead of `or`). The user's input is used in the attempt at the condition in the selection structure (so it does get **Mark D**).

## Response 4

```
character = input()
if character >= "a" and <= "z":
  print("LOWER")
else:
  print("NOT LOWER")
```

**Marks awarded: 6**

This response gets **Mark C** as a meaningful variable name has been used and **Mark E** as data is obtained from the user and stored in a variable. There is an attempt at a selection structure so **Mark B** is awarded and also **Mark G** as the two messages are in logically sensible places.

There is an attempt at checking if the character is lower case or not so it gets **Mark A** but it does not get **Mark F** as while the syntax is correct the logic of the program is not (2nd condition does not specify what "z" is being compared to). The user's input is used in the attempt at the condition in the selection structure (so it does get **Mark D**).

## Response 5

```
character = input()
if character >= "a" and character <= "z":
  print("LOWER")
else:
  print("NOT LOWER")
```

**Marks awarded: 7**

Fully correct answer.

## Question 15

A program has been written in Python to display all the odd integers between 1 and the largest odd number smaller than an integer entered by the user. The program is shown in **Figure 6**.

**Figure 6**

```
odd = 1
number = int(input("Enter an integer: "))
while odd != number:
  print(odd)
  odd = odd + 2
print("Finished!")
```

The program works correctly if the integer entered by the user is an odd, positive integer. For example, if 7 is entered the program correctly displays the values 1, 3 and 5

The program does not work correctly if an odd integer less than 1 is entered by the user. For example, when −7 is entered the program should display the values 1, -1, -3 and -5 but it does not do this.

Using Python only, change the program code inside the while loop so that it will work correctly for any odd integer entered by the user.

**[4 marks]**

## Mark scheme

| Question | Marking guidance |
|---|---|
| 15 | **4 marks for AO3 (refine)**<br><br>**<u>Program Logic</u>**<br><br>**Mark A:** for using a selection structure with else part or two selection structures (even if the syntax is incorrect)<br><br>**Mark B:** for correct condition(s) in selection statement(s) (even if the syntax is incorrect)<br><br>**Mark C:** for statement that subtracts two from odd under the correct conditions (even if the syntax is incorrect)<br><br>**Mark D:** for odd being output and doing one of adding or subtracting two but not both each time loop repeats (even if the syntax is incorrect)<br><br>**I.** while loop from question if included in answer<br><br>**I.** case of program code<br><br>**Maximum 3 marks** if any errors in code. |

## Response 1

```
print(odd)
odd = odd + 2
```

**Marks awarded: 0**

While both of these lines of code are needed for the answer to be correct, this is just the original (unchanged) code that was in the `while` loop given in the question so it does not get **Mark D** as there needs to have been a sensible attempt at answering the question.

## Response 2

```
print(odd)
odd = odd - 2
```

**Marks awarded: 1**

While not much better than the answer given in Response 1 it does achieve **Mark D** as the code has been modified and each time the loop repeats it will change the value of `odd` by two and it does display the value of `odd`.

## Response 3

```
print(odd)
if odd < 2:
  odd = odd - 2
odd = odd + 2
```

**Marks awarded: 0**

This response does not get **Mark A** as the selection structure does not contain an `else` part. The condition is incorrect (uses `odd` instead of `number`) so it does not get **Mark B**. It does not subtract two under the correct conditions and each time the loop repeats, and `odd` has a value less than 2, both the addition and subtraction statements will be executed.

## Response 4

```
if number < 0:
  odd = odd - 2
else:
  odd = odd + 2
print(odd)
```

**Marks awarded: 3**

This response meets the criteria for all four mark points but only gets 3 marks as there is a logical error in the code – the program displays the value of `odd` after changing it rather than before, this means the value 1 will not be displayed.

## Response 5

```
print(odd)
if number < 0:
odd = odd - 2
else:
odd = odd + 2
```

**Marks awarded: 4**

This response has all the correct lines of code in the required order. However, the indentation is incorrect which could affect the logic flow of the code. In this case, as there is only a single line of code in each part of the selection statement the logic is unaffected and so full marks can be awarded. If there had been more lines of code within the selection structure then marks would have been lost because of the lack of indentation.

## Response 6

```
print(odd)
if Number < 0:
  odd = odd - 2
else:
  odd = odd + 2
```

**Marks awarded: 4**

This response meets the criteria for all four mark points. The case of `number` is incorrect but case is ignored in responses to programming questions as the quality of a student's handwriting may not make it obvious which case has been used.

## Question 18

Write a Python program that calculates an estimate of the braking distance in metres for a new model of go-kart that is travelling between `10` and `50` kilometres per hour (kph).

Your program should:

- keep asking the user to enter a speed for the go-kart until they enter a speed that is between `10` and `50` (inclusive)
- calculate the braking distance in metres by dividing the speed by 5
- ask the user if the ground is wet (expect the user to enter `yes` if it is)
- if the ground is wet, multiply the braking distance by 1.5
- output the final calculated braking distance.

You **should** use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent you code accurately.

**[8 marks]**

## Mark scheme

| Question | Marking guidance |
|---|---|
| 18 | **2 marks for AO3 (design) and 6 marks for AO3 (program)**<br><br>**Program Design**<br><br>**Mark A** for using an iterative structure to validate the user input of speed (even if logic is incorrect);<br><br>**Mark B** for using meaningful variable names **and** suitable data types throughout (speed can be real or integer, breaking distance must be real, the `IsWet` input must be string);<br><br>**Program Logic**<br><br>**Mark C** for getting user input for both the speed and `IsWet` in appropriate places;<br><br>**Mark D** for using a WHILE loop or similar to re-prompt for the user input (even if it would not work);<br><br>**Mark E** for using a correct Boolean condition with the validation structure;<br><br>**Mark F** for calculating the braking distance correctly (i.e. divided by 5);<br><br>**Mark G** for using a selection structure to adjust the braking distance calculation if the user input required it (even if it would not work);<br><br>**Mark H** for outputting the braking distance in a logically correct place;<br><br>**I.** Case of program code<br><br>**Maximum 7 marks** if any errors in code. |

## Response 1

```
if speed < 10 or speed > 50:
  speed = int(input())
braking_distance = speed / 5
IsWet = input()
if IsWet:
  braking_distance = braking_distance * 1.5
print(braking_distance)
```

**Marks awarded: 5**

This response has used a selection structure to validate speed instead of an iterative structure, this means it does not get **Mark A** or **Mark D**. However, the condition on the structure used to validate `speed` is correct so **Mark E** is awarded. **Mark C** is not awarded as the code will never get the user to enter the `speed` as the only time this is done is inside the selection structure used for validating `speed`. The student has got the value of `IsWet` from the user but both inputs are needed for **Mark C** to be awarded.

All variables used have sensible identifiers and data types so **Mark B** is awarded. They have given the braking distance a value of `speed` divided by five so get **Mark F**.

**Mark G** awarded even though the code is incorrect, due to the condition assuming that `IsWet` is a Boolean variable rather than a string, as the mark scheme states that this mark can be awarded even if it would not work. **Mark H** is also awarded.

## Response 2

```
speed = int(input())
while speed < 10 or speed > 50:
  speed = int(input())
braking_distance = speed / 5
IsWet = input()
if IsWet == "yes":
  speed = speed * 1.5
print(braking_distance)
```

**Marks awarded: 7**

This response has a fully correct iterative structure to validate speed so it gets **Mark A**, **Mark D** and **Mark E**. Both required inputs are obtained from the user in appropriate places in the code so **Mark C** is awarded.

All variables used have sensible identifiers and data types so **Mark B** is awarded. The student has given the braking distance a value of speed divided by 5 so gets **Mark F**.

**Mark G** is not awarded as the error in the code inside the `if` statement means that `speed` and not `braking_distance` is adjusted; the student does get **Mark H** as their calculated value is displayed.

## Response 3

```
speed = int(input())
repeat not(speed >= 10 and speed <= 50):
  speed = int(input())
braking_distance = speed / 5
IsWet = input()
if IsWet == "yes":
  braking_distance = braking_distance * 1.5
print(braking_distance)
```

**Marks awarded: 7**

This response has an iterative structure to validate speed but the syntax of the loop (`repeat`) is not correct Python syntax. However, **Mark A** can still be awarded as it does not depend on the iterative structure being correct. The conditions are logically equivalent to those used on the mark scheme. Both required inputs are obtained from the user in appropriate places in the code so **Mark C** is awarded.

All variables used have sensible identifiers and data types so **Mark B** is awarded. The student has given the braking distance a value of `speed` divided by five so gets **Mark F**.

**Mark G** and **Mark H** are both awarded.

As there is an error in the code that is more than a minor syntax error only 7 marks are awarded.

## Response 4

```
speed = int(input())
while speed < 10 or speed > 50:
  speed = int(input())
braking_distance = speed / 5
IsWet = input()
if IsWet == "yes:
  braking_distance = braking_distance * 1.5
print(braking_distance)
```

**Marks awarded: 8**

There is a minor syntax error on line 6 (missing speech mark) but this would not prevent the awarding of full marks.

Full marks awarded.

# Get help and support

Visit our website for information, guidance, support and resources at aqa.org.uk/8525

You can talk directly to the Computer Science subject team

E: computerscience@aqa.org.uk

T: 0161 957 3980

aqa.org.uk



G02235