ANSWERS

A1.2 Operations, strings and random numbers

Programming techniques and object-oriented programming · A level · AQA 7517 4.1.1.3, Eduqas A500QS 1.4 · about 20 min

What this lesson is about

Integer division and MOD with negatives, rounding and truncation, XOR, string and date conversions, and pseudo-random numbers.

- Arithmetic
- Relational and Boolean operations
- String handling
- Random number generation

Questions

6 marks in all

1. What does this program print? [1 mark]

```
print(-17 // 5, -17 % 5)
```

Answer:

```
-4 3
```

Python's integer division rounds down to -4, and the remainder takes the divisor's sign: 5 * -4 + 3 is -17.

2. What does this program print? [1 mark]

```
import math
x = -2.6
print(round(x), int(x), math.floor(x))
```

Answer:

```
-3 -2 -3
```

Rounding gives the nearest whole number, -3; truncation removes the fraction, -2; floor rounds down, -3.

3. What is True XOR True? [1 mark]

- ☐ **A** False
- ☐ **B** True

Answer: A. XOR is true only when exactly one input is true.

4. What does this program print? [1 mark]

```
s = "R135,F20"
comma = s.find(",")
print(comma, int(s[1:comma]) + 1, s[comma + 1:] + "!")
```

Answer:

```
4 136 F20!
```

The comma is at index 4, the substring from index 1 to before 4 is "135", and the rest of the string is "F20".

5. What does this program print? [1 mark]

```
heading = 30
for turn in [-90, -45, 200]:
    heading = (heading + turn) % 360
    print(heading)
```

Answer:

```
300
255
95
```

30 - 90 is -60, which MOD 360 gives 300; then 255; then 455 MOD 360 is 95.

6. Why does setting the same seed before generating numbers give the same sequence each time? [1 mark]

☐ **A** The generator is pseudo-random: each number is calculated from the previous state, starting from the seed

☐ **B** The computer stores every random number it has ever made

☐ **C** The seed stops the numbers being random at all times

☐ **D** Random numbers always repeat after one use

Answer: A. A pseudo-random generator is deterministic: the same starting state gives the same sequence.

The task: drive a command string

A route arrives as one string of commands separated by commas: `COMMANDS = "F20,L90,F15,R135,B5"`. Each command is a letter and a whole number: `F` drives forward that many cm, `B` drives backward that many cm, `R` turns right that many degrees and `L` turns left that many degrees. Go through the string and carry out each command on the robot. Keep two totals as you go: the distance driven in cm (forward and backward both add) and the heading in degrees, starting at 0, turning right adding and turning left subtracting, always kept from 0 to 359 with MOD 360. Convert each amount with `int`. At the end print `driven <cm> cm, heading <degrees>`, for this string `driven 40 cm, heading 45`. Work both totals out from the string; do not type them.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

COMMANDS = "F20,L90,F15,R135,B5"

for command in COMMANDS.split(","):
    print(command)
```

The hint students can ask for: Each command is one letter followed by a number. Separate the commands, then take the first character as the letter and everything after it as the amount. Distances add up whichever way you drive; turns change the heading, and MOD keeps it in range.

A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

COMMANDS = "F20,L90,F15,R135,B5"

driven = 0
bearing = 0
for command in COMMANDS.split(","):
    letter = command[0]
    amount = int(command[1:])
    if letter == "F":
        forward(50, distance=amount)
        driven = driven + amount
    elif letter == "B":
        backward(50, distance=amount)
        driven = driven + amount
    elif letter == "R":
        turn_right(30, angle=amount)
        bearing = (bearing + amount) % 360
    elif letter == "L":
        turn_left(30, angle=amount)
        bearing = (bearing - amount) % 360
print(f"driven {driven} cm, heading {bearing}")
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.