ANSWERS

A1.5 Scope, lifetime and debugging in an IDE

Programming techniques and object-oriented programming · A level · OCR H446 2.2.1, AQA 7517 4.1.1.13, Eduqas A500QS 1.8 · about 20 min

What this lesson is about

Local and global variables, why locals are good practice, and breakpoints, stepping, watches and tracebacks.

- Scope and lifetime
- Shadowing, and changing a global
- Why local variables are good practice
- Debugging tools

Questions

6 marks in all

1. What is the lifetime of a local variable? [1 mark]

- ☐ **A** From when its subroutine is called until the subroutine returns
- ☐ **B** The whole time the program runs
- ☐ **C** Until the variable is printed
- ☐ **D** Until the next global variable is declared

> **Answer:** A. A local is created on each call and destroyed when that call ends.

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

```
speed = 80

def slow():
    speed = 30
    return speed

print(slow(), speed)
```

> **Answer:**
>
> ```
> 30 80
> ```
>
> The assignment inside slow makes a new local speed; the global speed is still 80.

3. Why is it good practice to use local variables rather than global variables in subroutines? [1 mark]

Tick every answer that is true.

- ☐ **A** The subroutine can be tested and reused on its own
- ☐ **B** A value cannot be changed unexpectedly by another part of the program
- ☐ **C** The same name can be used safely in different subroutines
- ☐ **D** Local variables can be used anywhere in the program

> **Answer:** A, B, C. Locals keep subroutines independent; being usable anywhere describes globals.

4. A program gives the wrong total, but only on the 40th time round a loop. Which IDE feature lets you run at full speed and then pause at the line that updates the total? [1 mark]

- ☐ **A** A breakpoint
- ☐ **B** Syntax highlighting
- ☐ **C** Auto-completion
- ☐ **D** Code folding

> **Answer:** A. A breakpoint pauses the run at a chosen line, so you can inspect variables there.

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

```
total = 0

def add(n):
    global total
    total = total + n

add(5)
add(7)
print(total)
```

> **Answer:**
>
> ```
> 12
> ```
>
> global makes total inside add refer to the global variable, so both calls change it.

6. An exception is not handled and Python prints a traceback. What does the traceback show? [1 mark]

- ☐ **A** The chain of subroutine calls that led to the line where the exception was raised
- ☐ **B** Every variable's value
- ☐ **C** The lines that ran successfully
- ☐ **D** A suggested fix

> **Answer:** A. Read it to see which call passed the bad value to the line that failed.

The task: fix the scope bug

This program should creep towards the wall in 5 cm steps while `distance()` is more than 25, then print `steps: <n>`, the number of steps taken. It stops with an `UnboundLocalError`. Run it and read the message first. Fix it **without** `global`: give `step` a parameter, `count` (the number of steps so far, an integer), and make it return `count + 1`, with the main program assigning the result back. Add a watch line: before each move, `step` prints `step <n>: <cm> cm`, where `<n>` is the number of the step about to be taken (1 for the first) and `<cm>` is the value `distance()` returns.

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

steps = 0

def step():
    forward(50, distance=5)
    steps = steps + 1

while distance() > 25:
    step()
print("steps:", steps)
```

The hint students can ask for: Read the error message: it names the variable and says it is local. The function only needs the count in and the new count out, so give it a parameter and a return value, and let the caller keep the result.

A solution

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

def step(count):
    print(f"step {count + 1}: {distance()} cm")
    forward(50, distance=5)
    return count + 1

steps = 0
while distance() > 25:
    steps = step(steps)
print("steps:", steps)
```

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