WORKSHEET

A1.1 Data types and programming constructs

Programming techniques and object-oriented programming · A level · OCR H446 2.2.1, AQA 7517 4.1.1.1, Eduqas A500QS 1.4 · about 20 min

Name _____ Class _____ Date _____

What this lesson is about

The A level data types, references and user-defined types, constants, and definite and indefinite iteration.

- The types you need to know
- References: two names, one value
- User-defined types
- Declaration, assignment and constants
- Iteration: definite and indefinite

Questions

6 marks in all

1. A variable holds the location in memory where a list of readings is stored, rather than the readings themselves. What is its data type? [1 mark]

- ☐ **A** Pointer/reference
- ☐ **B** Array
- ☐ **C** Record
- ☐ **D** Integer

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

```
a = [1, 2]
b = a
b.append(3)
c = list(a)
c.append(4)
print(a, len(c))
```

3. A loop has its condition tested at the end. What is always true of it? [1 mark]

- ☐ **A** Its body runs at least once
- ☐ **B** Its body may run zero times
- ☐ **C** It is definite iteration
- ☐ **D** It cannot use a Boolean condition

4. Which are advantages of using a named constant such as SAFE_CM instead of typing 25 wherever it is needed? [1 mark]

Tick every answer that is true.

- ☐ **A** The value only has to be changed in one place
- ☐ **B** The name explains what the value means
- ☐ **C** It cannot be changed by mistake while the program runs, in a language that enforces constants
- ☐ **D** It makes the program run in less memory

5. A robot must repeat 'drive 5 cm' until the wall is 25 cm away. The number of repeats is not known in advance. What kind of iteration is this? [1 mark]

- ☐ **A** Indefinite iteration
- ☐ **B** Definite iteration
- ☐ **C** Recursion
- ☐ **D** Selection

6. Python has no character type. What type does it use to store a single character such as "F"? Give the Python type name. [1 mark]

The task: readings as a record type

Define a record type `Reading` with `namedtuple` and three fields: `heading` (an integer, 0, 90, 180 or 270), `cm` (the real number `distance()` returns) and `clear` (a Boolean, `True` when `cm` is more than the constant `CLEAR_CM`). Declare `CLEAR_CM = 40` as a constant and use the name, not `40`, in the comparison. The robot starts facing heading 0. Take a reading, store it in an array, turn right 90 degrees, and repeat until there are four readings and the robot faces its starting direction. Then print each record with `print(r)`, which shows `Reading(heading=90, cm=51.0, clear=True)`, and finally print `clear directions: <n>`, the number of records whose `clear` field is `True`.

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

from collections import namedtuple

Reading = namedtuple("Reading", ["heading", "cm", "clear"])

survey = []
# take four readings, one every 90 degrees

print("clear directions:", 0)
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a1-1-data-types-and-constructs/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Add a fourth field, `taken`, holding the time of the reading from `clock()`. What type is it?
2. Rewrite the survey loop as a condition-at-the-end loop that stops when the robot has turned a full circle.
3. Try `r.cm = 0` on one of your records. What happens, and why might that be useful for sensor data?