WORKSHEET

A13.4 Partial application and composition

Functional programming · A level · AQA 7517 4.12.1.4, Eduqas A500QS 1.4 · about 25 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Why every Haskell function takes one argument, partial function application, functools.partial, and composing functions into a sensor pipeline.

- Every function takes one argument
- Partial application in Python
- Composition of functions
- A sensor pipeline

Questions

6 marks in all

1. In Haskell, mult x y = x * y and triple = mult 3. What is triple 7? [1 mark]

2. add :: Integer -> Integer -> Integer. What is the type of add 4? [1 mark]

- ☐ **A** Integer -> Integer
- ☐ **B** Integer
- ☐ **C** Integer -> Integer -> Integer
- ☐ **D** (Integer, Integer) -> Integer

3. f(x) = x + 2 and g(y) = y^3. What is (g ∘ f)(2)? [1 mark]

4. f(x) = x + 2 and g(y) = y^3. What is (f ∘ g)(2)? [1 mark]

5. f: A → B and g: B → C. What is the type of g ∘ f? [1 mark]

- ☐ **A** A → C
- ☐ **B** C → A
- ☐ **C** B → B
- ☐ **D** A → B → C

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

```
def compose(g, f):
    return lambda x: g(f(x))

double = lambda x: x * 2
less5 = lambda x: x - 5
print(compose(double, less5)(10), compose(less5, double)(10))
```

The task: calibrate by composition

Build a calibration pipeline for the distance sensor from small curried functions: - `scale(factor)` returns a function of one number `cm` that gives `cm * factor`. - `offset(amount)` returns a function of one number `cm` that gives `cm + amount`. - `compose(g, f)` returns a function of one value `x` that gives `g(f(x))`. Using only those three, make `pipeline`: first `offset(-2)`, then `scale(10)`. Apply it to each reading in `RAW = (38.0, 51.5, 20.0)` and print the results on one line as `mm: 360.0 495.0 180.0`. Then compose the same two stages in the **wrong** order and print its results the same way as `wrong order: <three numbers>`. Round each result to one decimal place. The robot does not move.

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

RAW = (38.0, 51.5, 20.0)

def pipeline(cm):
    return cm * 10 - 2

print("mm:", pipeline(RAW[0]))
```

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

Do it on the robot

www.bugbotlab.com/learn/a13-4-partial-application-and-composition/

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

Challenges

1. `partial(clamp, 0, 60)` fixes `low` and `high`. Can `partial` fix only `high`? Try `partial(clamp, high=60)` and explain what happens.
2. Write `compose_all(functions)` that composes a whole list of stages, applying the first in the list first.
3. Give the types of `scale`, `scale(10)` and `scale(10)(3.5)` in Haskell-style notation.