WORKSHEET

A2.5 Procedural, functional and data abstraction

Recursion and computational thinking · A level · OCR H446 2.1.1, AQA 7517 4.4.1.4 · about 20 min

Name _____ Class _____ Date _____

What this lesson is about

Information hiding, hiding the values, the method and the representation, and swapping a data type's insides.

- Information hiding
- Procedural abstraction
- Functional abstraction
- Data abstraction

Questions

6 marks in all

1. What is information hiding? [1 mark]

- ☐ **A** Hiding all details of an object that do not contribute to its essential characteristics
- ☐ **B** Encrypting data so it cannot be read
- ☐ **C** Removing comments from code
- ☐ **D** Storing data in a hidden file

2. print(30 * 20) is replaced by def area(width, height): return width * height. Which kind of abstraction is this step? [1 mark]

- ☐ **A** Procedural abstraction: the actual values are abstracted away, leaving a computational method
- ☐ **B** Functional abstraction: the method is hidden
- ☐ **C** Data abstraction: the representation is hidden
- ☐ **D** Abstraction by generalisation

3. Two functions both return the smallest reading: one scans the list, one sorts it. A caller uses either without knowing which. What does this show? [1 mark]

- ☐ **A** Functional abstraction: the particular computation method is disregarded
- ☐ **B** Procedural abstraction
- ☐ **C** Decomposition
- ☐ **D** Composition

4. A stack is used only through push, pop and is_empty. Inside, it could be an array with a top pointer or a linked list. What is this? [1 mark]

- ☐ **A** Data abstraction
- ☐ **B** Procedural decomposition
- ☐ **C** Problem reduction
- ☐ **D** Representational abstraction

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

```
def new_time(minutes, seconds):
    return minutes * 60 + seconds

def add_seconds(t, s):
    return t + s

def show(t):
    return f"{t // 60}:{t % 60:02d}"

lap = add_seconds(new_time(2, 45), 30)
print(show(lap))
```

6. What are benefits of hiding a component's implementation behind its interface? [1 mark]

Tick every answer that is true.

- ☐ **A** The inside can be changed without breaking code that uses it
- ☐ **B** Users have less to understand
- ☐ **C** Nobody can depend on an internal detail by accident
- ☐ **D** The component always runs faster

The task: swap the insides

The robot stands in the middle of the mat, turns through eight 45 degree steps and logs a distance at each. The starter's log is a list, and the program at the bottom uses it only through five operations: `new_log()`, `add_reading(log, cm)`, `count(log)`, `mean(log)` and `nearest(log)`. Change the **representation** without changing the program at the bottom: - a log must now be a dictionary holding only three figures: how many readings there have been, their running total, and the smallest reading so far; - `new_log()` returns an empty log; `add_reading(log, cm)` updates the figures for a reading `cm` (a number of cm) and returns nothing; `count(log)` returns the number of readings; `mean(log)` returns the mean reading; `nearest(log)` returns the smallest reading; - do not keep the readings, and do not use `append`, `len` or `sum` anywhere. The program at the bottom must still print `count 8`, then `mean <m>` and `nearest <n>` with exactly the values the list version prints.

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

def new_log():
    return []

def add_reading(log, cm):
    log.append(cm)

def count(log):
    return len(log)

def mean(log):
    return sum(log) / len(log)

def nearest(log):
    return min(log)

# the program that uses the log: do not change anything below this line
log = new_log()
for i in range(8):
    add_reading(log, distance())
    turn_right(40, angle=45)
print("count", count(log))
print("mean", round(mean(log), 1))
print("nearest", nearest(log))
```

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

Do it on the robot

www.bugbotlab.com/learn/a2-5-procedural-functional-and-data-abstraction/

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

Challenges

1. Add a sixth operation, `farthest(log)`. What must change in the representation, and what must not change in the program at the bottom?
2. The dictionary version cannot give the median reading. Explain why, and what that says about choosing a representation.
3. Is `sorted` a procedural or a functional abstraction for someone who calls it? Justify your answer.