Unit 1 · Objects and Classes

ANSWER KEY · FOR TEACHERS

Full marks 33

Give the marks for the points in the marking scheme; an answer in other words that says the same is right.

1 Key words

1. class
2. object
3. attribute
4. method
5. constructor
6. encapsulation
7. inheritance
8. composition

Each carries 0.5 mark. **K · 4**

2 Label the diagram

1. riders
2. board(n)
3. made from
4. micro : Bus

Each carries 0.5 mark. **U · 2**

3 Multiple choice

i. **b.** the object the method was called on **K · 1**

`sajha.board(25)` runs as `Bus.board(sajha, 25)`, so self is sajha's object. Called on micro, self is micro's object.

ii. **a.** 0 **A · 1**

`a` and `b` are two separate objects made from the same class, even though their values are the same. Each has its own `riders`. Boarding `a` changes only `a`'s. (Python doesn't know that number plates should be different; if that matters, the program must check it.)

iii. **c.** AttributeError: the object has no attribute marks, because `marks = []` only made a local variable of `__init__`. **A · 1**

Without `self.`, the name is local to `__init__`, thrown away when it ends. The object never gets the attribute, so `self.marks` fails with an AttributeError (a NameError is for a plain name, not one after a dot).

iv. **b.** Microbus and Vehicle **U · 1**

"A microbus is a kind of vehicle" is true, and everything a vehicle can do, a microbus can. The others are "has a": a library has books, a route has stops, a school has students. They are joined by composition.

v. **c.** `Savings.rate = 0.07` **A · 1**

The rate is a class attribute, so change it through the class's name, and every account sees the new rate. `sita.rate = 0.07` and `self.rate = 0.07` only give one account a rate of its own, which hides the shared one; `rate = 0.07` in a method makes a local variable, gone when the method ends.

4 What does it print?

a. **MARKING SCHEME** **A · 2**

- ✓ `50` for `a.sold`: line 10 gave the object **a** an attribute of its own, which hides the class's.
- ✓ `3 3`: the class attribute counts all three tickets, and **b**, with no `sold` of its own, reads the class's.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

b. MARKING SCHEME A · 3

- ✓ `goat at 3`: Goat has no `show()`, so Piece's runs, and its `self.name()` finds Goat's own `name()`.
- ✓ `TIGER piece at 0`: Tiger's own `show()`, which calls Piece's through `super()`; Tiger has no `name()`, so Piece's gives `piece`.
- ✓ `piece at 7`, from a plain Piece.

5 Find and fix the bug

MARKING SCHEME A · 3

- ✓ The list on line 2 belongs to the class: there is only one, and every student shares it.
- ✓ `self.marks = []` inside `__init__` (and line 2 removed), so each object gets its own list.
- ✓ `Asha [72]`.

6 Trace the program

ANSWER A · 4

Call	`_balance`	`refused`	Output
(before)	0	0	
`sita.deposit(500)`	500	0	
`sita.deposit(-100)`	500	0	
`sita.deposit(300)`	800	0	
`hari.withdraw(1000)`	800	1	
`hari.withdraw(250)`	550	1	
			550 1

`hari = sita` makes a second name for the same account, not a second account, so Hari's withdrawals come out of Sita's balance.

MARKING SCHEME

- ✓ The deposits: 500, then no change for −100 (refused by `amount > 0`), then 800.
- ✓ `withdraw(1000)` refused: the balance stays 800 and `refused` becomes 1.
- ✓ `withdraw(250)` taken from the same account, because **hari** and **sita** are one object: 550.
- ✓ Output `550 1`, once, at the end.

7 Short answers

a. MARKING SCHEME U · 2

- ✓ i. Inheritance (Ambulance a child class of Vehicle): an ambulance is a kind of vehicle, and can do everything a vehicle can.
- ✓ ii. Composition (School holds a list of Classroom objects): a school has classrooms; a classroom is not a kind of school.

b. MARKING SCHEME U · 2

- ✓ Polymorphism: the same call, `p.show()`, runs a different method depending on each object's class.
- ✓ The loop doesn't need to check which kind each piece is; a new kind of piece works in it without the loop changing.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

8 Write a program

ANSWER **A · 4**

```
class Hostel:
    def __init__(self, name, beds):
        self.name = name
        self.beds = beds
        self.residents = []

    def admit(self, person):
        if len(self.residents) < self.beds:
            self.residents.append(person)
            return True
        return False

    def leave(self, person):
        if person in self.residents:
            self.residents.remove(person)

    def __str__(self):
        return f"{self.name}: {len(self.residents)} of {self.beds} beds taken"

h = Hostel("Girls' hostel", 2)
print(h.admit("Asha"), h.admit("Gita"), h.admit("Mina"))
h.leave("Asha")
print(h)
```

OUTPUT

```
True True False
Girls' hostel: 1 of 2 beds taken
```

MARKING SCHEME

- ✓ `__init__` with `self.name`, `self.beds` and a new list `self.residents = []` for each hostel.
- ✓ `admit()` checks for a free bed before adding, and returns True or False.
- ✓ `leave()` removes the person (checking they are there, or with `remove`).
- ✓ `__str__` returns the text with the number of residents and beds, and the objects are made and used as asked.

9 Think

ANSWER **HA · 2**

The class can change how it keeps the balance, to a passbook list of deposits and withdrawals whose total is the balance, and only the class changes: `balance()` now adds up the passbook, and every piece of code that calls it carries on as before. Code that read `_balance` directly would all have to be found and changed. The underscore is an agreement between programmers, not a lock: it says the attribute is the class's own business, so other code should leave it alone and use the methods, which check every change.

MARKING SCHEME

- ✓ Only the class changes when the way the data is kept changes (`balance()` adds up the passbook), because other code uses only the method.
- ✓ The underscore marks the attribute as the class's own, by agreement: Python doesn't protect it, but programmers leave it alone.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

10 Activity: Mistakes with classes

(not marked)

WHAT STUDENTS SHOULD FIND

- Program 1: line 8, **AttributeError** (the Bus object has no attribute riders). The fix is on line 5: `self.riders = 0`, not `riders = 0`, which only made a local variable of `__init__`.
- Program 2: line 11, the call, **TypeError** (board takes 1 positional argument but 2 were given). The fix is on line 7: `def board(self, n):`, because the call passes the object as the first argument.
- Program 3: line 7, **TypeError** (missing 1 required positional argument: seats). The fix is on line 7: `micro = Bus("Ba 3 Ja 118", 14)`.

Marks by level	Knowledge	Understanding	Application	Higher ability	**Total**
Marks	5	7	19	2	**33**

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

