

## 6.4 Functions and file processing

Name: \_\_\_\_\_ Class: \_\_\_\_\_ Date: \_\_\_\_\_ Total: 22 marks

**KEY WORDS** function      • parameter      • return      • local variable  
• global variable      • mode

### Objective

Build the skills to answer exam questions on **functions and file processing** – parameters, return values, local and global scope, and reading and writing text files.

You must be able to:

- write a **function** with **parameters** that **returns** a value
- tell a **local variable** from a **global variable**
- choose the file **mode** "r", "w" or "a", and say what each does to an existing file
- read a file line by line, split each line into fields and process them

### 1 Key knowledge and worked examples

Study these first. Each one shows the method for a question type used later.

#### ■ Key knowledge: functions and scope

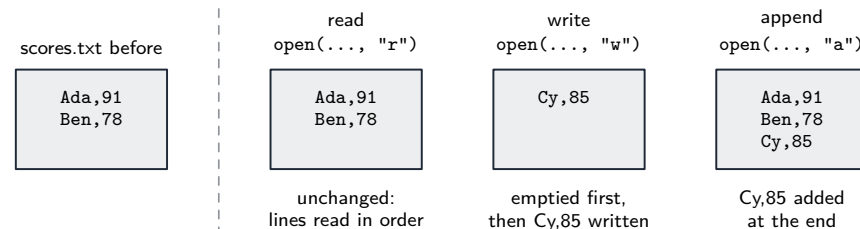
```
RATE = 0.1                                # global constant

def discount(price, qty):
    total = price * qty                    # local: exists only during the call
    if qty >= 10:
        total = total - total * RATE
    return total

print(discount(4.0, 12))                  # 43.2
```

- **price** and **qty** are parameters; 4.0 and 12 are the **arguments** passed in.
- **return** sends the result back to the caller; a function with no **return** gives **None**.
- **total** cannot be used outside **discount**: it is local. Splitting a program into functions is **modularity**.

#### ■ Key knowledge: file modes



- with `open(name, mode)` as `f`: closes the file when the block ends, even after an exception.
- Reading a missing file raises `FileNotFoundError`.

#### ■ Worked example: processing a file

`scores.txt` holds one `name,score` per line. Print the names with a score of at least 80, and the mean score.

```
count, total = 0, 0
with open("scores.txt", "r") as f:
    for line in f:
        name, score = line.strip().split(",")
        score = int(score)
        count = count + 1
        total = total + score
        if score >= 80:
            print(name)
print(total / count)
```

1. `for line in f` reads one line at a time, "Ada,91\n".
2. `strip()` removes the "\n"; `split(",")` gives ["Ada", "91"].
3. `int(score)` casts the text to a number before comparing or adding.
4. For Ada,91 and Ben,78: prints Ada, then  $(91 + 78)/2 = 84.5$ .

### 2 Practice

**2.1** Write a function `area(w, h)` that returns the area of a rectangle, and a statement that prints the area of a 3 by 5 rectangle. [2]

---

---

---

**2.2** In the `discount` code above, name (i) a parameter; (ii) a local variable; (iii) a global constant. [3]

---

---

---

**2.3** State the mode to use for each, and why: (i) adding today's temperature to a log kept all year; (ii) loading saved settings; (iii) replacing last week's report with a new one. [3]

---

---

---

**2.4** `line = "Lee,17,B\n"`. State the value of `line.strip().split(",")`. [1]

---

### 3 Exam-style questions

**3.1** What does `open("data.txt", "w")` do to an existing `data.txt`? [1]

- ☐ **A** adds new lines to the end  
☐ **B** empties it before writing  
☐ **C** reads it from the start  
☐ **D** raises an error

**3.2** `def f(x): y = x * 2`. What does `print(f(3))` print? [1]

- ☐ **A** 6  
☐ **B** 3  
☐ **C** None  
☐ **D** an error, as `y` is undefined

**3.3** Which is the main advantage of using functions? [1]

- ☐ **A** the program uses no variables  
☐ **B** each part can be written, tested and reused separately  
☐ **C** the program runs without errors  
☐ **D** files are closed automatically

**3.4** A club stores its members in `members.txt`, one line each, as `name,year` joined, for example `Mei,2019`.

(a) Write a function `joined_before(filename, year)` that returns how many members joined before `year`. [5]

---

---

---

---

---

---

---

---

(b) The file holds `Mei,2019`, `Tom,2023` and `Raj,2015`. State the value of `joined_before("members.txt", 2020)`. [1]

---

(c) Write a statement that adds the new member `Zoe,2026` to the file without losing the others. [2]

---

---

(d) Explain what happens if the file does not exist, and how the program could deal with it. [2]

---

---

## Solutions

---

Each answer shows the steps, then the **result**.

### 2.1

- `def area(w, h):` with `return w * h`
- `print(area(3, 5))`

### 2.2

- (i) price or qty
- (ii) total
- (iii) RATE

### 2.3

- (i) **"a"**: keeps the earlier readings
- (ii) **"r"**: the file is only read
- (iii) **"w"**: the old contents should go

### 2.4

- `["Lee", "17", "B"]`, all strings

### 3.1 B

- write mode empties an existing file first

### 3.2 C

- `f` has no `return`, so it returns `None`

### 3.3 B

- modularity

### 3.4

(a) One answer:

```
def joined_before(filename, year):
    count = 0
    with open(filename, "r") as f:
        for line in f:
            name, joined = line.strip().split(",")
            if int(joined) < year:
                count = count + 1
    return count
```

- opens the file in read mode
- loops over the lines
- strips and splits each line
- casts the year and compares with `<`

- counts and returns the count after the loop

(b)

- Mei and Raj: **2**

(c)

- `with open("members.txt", "a") as f:`
- `f.write("Zoe,2026\n")`, with the newline

(d)

- `open` raises a `FileNotFoundError` and the program stops
- put the code in `try ... except FileNotFoundError` and print a message or return 0