Exercise walk-through

6.4
Functions and file processing

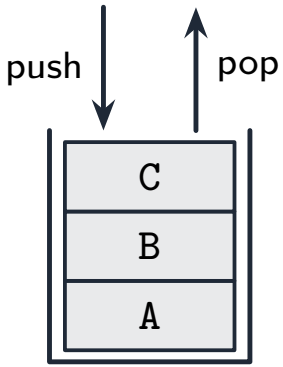
International Baccalaureate · IB Diploma · Computer Science · SL

Catching bad input with try and except

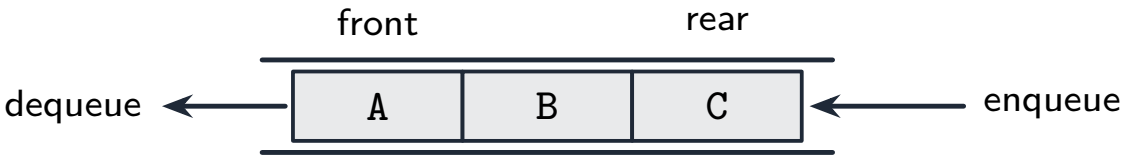
- An unhandled **exception** stops the program.
- The loop repeats until the cast succeeds.

```
while True:
    text = input("Quantity: ")
    try:
        qty = int(text)
        break
    except ValueError:
        print("Enter a whole number.")
```

Stacks and queues



stack: last in, first out



queue: first in, first out

> Stack: push and pop at the top (undo, back button). Queue: enqueue at the rear, dequeue at the front (print jobs).

Check yourself

Your turn

Exam habits to check before you answer.

A. Does your trace table give a new row each time a variable changes?

B. Did you state that Python indexes from 0?

C. Did you name a stack or queue's order, its two operations and a real use?

D. Did you open files with with and handle bad input with try and except?

E. Did you say the list must be sorted, and give low, high and mid at every probe?

Exercises 2.1·2.2

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]

Exercises 2.1·2.2

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]

2.1

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

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

2.2

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

Exercises 2.3·2.4

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]

Exercises 2.3·2.4

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]

2.3

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

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

2.4

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

Exercises 3.1·3.2

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

Exercises 3.1·3.2

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

3.1 B

- write mode empties an existing file first

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

3.2 C

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

Exercise 3.3

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

Exercise 3.3

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

3.3 B

- modularity

Exercise 3.4 (a)

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]

Exercise 3.4 (a)

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]

(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

Exercise 3.4 (b)

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

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

Exercise 3.4 (b)

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

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

(b)

- Mei and Raj: **2**

Exercise 3.4 (c)

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

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

Exercise 3.4 (c)

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

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

(c)

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

Exercise 3.4 (d)

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

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

Exercise 3.4 (d)

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

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

(d)

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