



University of  
Zurich<sup>UZH</sup>

# Exercises

---

Philipp Gloor<sup>1</sup>

<sup>1</sup>University of Zurich

## General Rules

- Always use the function names provided in the exercise
- Always create for each exercise a new file!

## Exercise 1 i

- Write a Python script that
  - prints 1 if  $x > y$
  - prints 0 if  $x == y$
  - prints -1 if  $x < y$
- Use `input()` to receive user input

## Exercise 1 i

- Attention: `input()` stores the input as a string (not as a number)
- If the input is supposed to be a number (`int`, `float`) you need to convert it

```
first_number = input('Please enter a first number ')
first_number = int(first_number)
second_number = input('Please enter a second number ')
second_number = int(second_number)
result = first_number + second_number
print(str(result))
```

## Exercise 2

- Write a Python script that reads x, y, and z from ‘input’ and prints True if  $x \leq y \leq z$  and False otherwise

## Exercise 3

- Write a Python script that prints every character of the string in reverse order
- Use a while loop to do this

## Exercise 4

### FizzBuzz with a while loop

FizzBuzz Print "FizzBuzz" if number % 15 == 0

Fizz Print "Fizz" if number % 3 == 0

Buzz Print "Buzz" if number % 5 == 0

- Create a list with numbers from 0 to 100
- Write a script that, in a `while`-loop prints the first element of the list (according to the FizzBuzz rules) and then deletes it from the list
- The while loop ends once the list is empty

## Exercise 5

- Write a script that counts the number of words in a list that are at least as long as the specified word length.
- Use a for loop to do this
- Example:

```
words = ['apple', 'banana', 'cat', 'elephant', 'dog']
min_length = 4
count = 0
# Use a for loop to count words with length >= min_length
# Should print 3 (apple=5, banana=6, elephant=8)
```

## Exercise 6.1

- Write a function called `distance(x1, y1, z1, x2, y2, z2)` which computes the distance between point 1 ( $x_1, y_1, z_1$ ) and point 2 ( $x_2, y_2, z_2$ ) and returns the result
- Note:
  - $\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$
  - $x^2$  is represented by `x**2` in Python
  - The root of x is computed with `math.sqrt(x)`
  - Use the `import math` statement at the beginning of the file

## Exercise 6.2

- Write a function `volume_from_radius(radius)`, which calculates the volume of a sphere
- Note:
  - $\text{volume} = \frac{4\pi}{3} \cdot r^3$
  - Pi is `math.pi`

## Exercise 6.3

Continue in the same file as ex 6.1.

- Write a function `volume_from_points(x1, y1, z1, x2, y2, z2)`
- This function calculates the volume of a sphere whose radius is the distance between the points  $(x_1, y_1, z_1)$  and  $(x_2, y_2, z_2)$

## Exercise 7.1 i

- Write a function `calculate_mark(points, max_points)` which returns a grade in the Swiss grading scale

$$\text{mark} = \frac{\text{points} \cdot 5}{\text{max points}} + 1$$

- The function rounds the grade to the nearest 0.5
  - 5.66666 -> 5.5
  - 5.75 -> 6
- The function should accept strings as arguments
- Arguments should be therefore converted to floats

## Exercise 7.2 i

- Ask for max\_points via input() function
- Write code that in a while loop asks for points as long as the user does not enter "exit"
- The grade should be printed after each run

```
# Read max_points from input
while True:
    # Read points from input
    if points == 'exit':
        break
    # call calculate_mark function
    # print result
```

## Exercise 7.3 i

- Change your code that it additionally asks for a name
- A dictionary should now store the grade of each name
  - The name is a key, the grade the value
- As soon as the user enters "exit" the program should print the grades of all names before it exits

## Exercise 7.4 i

- Change your code in such a way that for each name it additionally outputs if the user has passed or failed
- $\text{Mark} \geq 4 \rightarrow \text{passed}$
- $\text{Mark} < 4 \rightarrow \text{failed}$

## Exercise 7.5 i

- Change your `in` such a way that the application outputs the average grade before it exits

## Exercise 8

- Write an application that generates a random number between 1 and 100
- import random
- random.randrange(min, max)
- The user makes a guess and enters a number. If the number is incorrect, the program outputs whether the entered number was too small or too large and allows the user to guess again.
- The application quits when the correct number is guessed
- The application should output how many user attempts have been made before it quits

## Exercise 9

- Implement the opposite of Task 8 so that the user thinks of a number and makes the computer guess
- The user provides feedback on whether the number is too high, too small, or correct
  - < (too low)
  - > (too high)
  - = (correct)
- How many steps does the computer need?

## Exercise 10 i

- Write an application which repeatedly asks for a name and phone number until the user enters “exit”
- Each name/telephone number pair should be stored as an entry in a dictionary
  - The names are the keys of the dictionary
  - The telephone numbers are the values of the dictionary
- As soon as the user enters “exit”, create a JSON string of the dictionary using the `json.dumps()` function and store the string in a file called `address_book.txt`

## Exercise 10 ii

- Extend your application so that it reads the address\_book.txt file when it starts
- Convert the JSON text into a dictionary again

```
import json
address_book_file = open('address_book.txt', 'r')
address_book_dict = json.load(address_book_file)
```

- Ask the user if he wants to add more names or not
- Let the user search for names in the dictionary and print out the according phone number

## Exercise 11 i

- Write an application which repeatedly asks for a name and phone number until the user enters “exit”
- Each name/telephone number pair should be stored as an entry in a dictionary
  - The names are the keys of the dictionary
  - The telephone numbers are the values of the dictionary
- As soon as the user enters “exit”, create a JSON string of the dictionary using the `json.dumps()` function and store the string in a file called `address_book.txt`

## Exercise 11 ii

- Extend your application so that it reads the address\_book.txt file when it starts
- Convert the JSON text into a dictionary again

```
import json
address_book_file = open('address_book.txt', 'r')
address_book_dict = json.load(address_book_file)
```

- Ask the user if he wants to add more names or not
- Let the user search for names in the dictionary and print out the according phone number