Algorithms & Data Structures

Week 3 -Functions

The first part of this exercise sheet consists of refactoring the code you have done in week 2. Refactoring means rewriting some element of the code to improve its readability, performance, and reusability. Write **all your code** in a file named week3exercises.py.

# Exercise 1:

For this exercise, you could reuse some of the code you wrote in week 2 for exercise 1.

A fruit company sells bananas for £3.00 a kilogram plus £4.99 per order for postage and packaging. If an order is over £50.00, the P&P is reduced by £1.50. Write a **function** order\_price(quantity) that takes an int parameter quantity representing the number of kilo of bananas for the order, and returns the cost of the order in pence (as an int).

# Exercise 2:

Again, you could use some of the code you have written in week 2 for exercise 2.

Write a **function** maximum\_heart\_rate(age) that takes the age of the person as parameter (an int) and returns the maximum heart rate for that person (as an int). The maximum heart rate is given by:

Write a second **function** training\_zone(age, rate) that takes the age (as an int) and rate (the heart rate as an int) from the person as parameters and returns a string representing one of the four possible training zones of a person based on his or her age and training heart rate, rate. The zone is determined by comparing rate with the person's maximum heart rate m:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Interval range** | | | | | **Training Zone** |
| rate | ≥ | 0.9 m |  |  | Interval training |
| 0.7 m | ≤ | rate | < | 0.9 m | Threshold training |
| 0.5 m | ≤ | rate | < | 0.7 m | Aerobic training |
| rate | < | 0.5 m |  |  | Couch potato |

To practice code reuse, the function training\_zone(age, rate) must call the function maximum\_heart\_rate(age).

# Exercise 3:

Again, you could use some of the code you have written in week 2 for exercise 4.

## Development

Implement a **function**:

is\_valid\_password(password,

min\_length,

has\_upper,

has\_lower,

has\_numeric)

that takes the following parameters in that order:

1. password: a password as a string
2. has\_upper: a Boolean. If True the password must contains at least one uppercase character
3. has\_lower: a Boolean. If True the password must contains at least one lowercase character
4. has\_numeric: a Boolean. If True the password must contains at least one numeric character

The function returns True if the password meets the following criteria:

1. at least min\_length characters long,
2. contains at least one uppercase character if has\_upper is True
3. contains at least one lowercase character if has\_lower is True
4. contains at least one numeric character if has\_numeric is True
5. Does not contain any special characters (that is non-alphanumeric like punctuation)

In addition, by default the function checks that a password has both uppercase and lowercase, has a minimum length of 8 and contains at least one numeric character.

## Testing

Design a series of tests using the assert statement to test the functionality of, and validity of your function.

# Exercise 4:

Write a **function** sum\_digits(number) to calculate and return the sum of the digits of a given whole number (an int NOT a string) given as a parameter. For example,

>>> print(sum\_digits(1234))

10

**Note:** There are two ways to approach the problem, the simplest one is to convert the parameter number into a string within the function and then solve the problem one character at a time. However, if you want to challenge yourself, try to approach the problem differently by not converting the parameter number into any other data type.

# Exercise 5:

Write a **function** pairwise\_digits(number\_a, number\_b) that takes two whole numbers represented by strings (not int) as parameters and returns a binary string where a character 1 is used if the digits at the same index are the same, a 0 otherwise. If the two strings have different lengths, the output should be padded with 0s on the right-hand side to match the length of the longest string. Examples are given in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Examples | 1 | 2 | 3 |
| Input A | ‘1213’ | 1213 | ‘12130’ |
| Input B | ‘2113’ | ‘1043567’ | ‘121’ |
| Output | ‘0011’ | ‘1001000’ | ‘11100’ |

## Testing

Design a series of tests using the assert statement to test the functionality of, and validity of your function.