Python Exercises

All exercises should be completed in Spyder unless specified – enjoy!

Exercise 1: Print "Hello World"

Exercise 1A: Using Spyder, create a file called *myfirst.py*

Exercise 1B: Print the sentence:
"A python ate my mother-in-law, what should I do?"

Exercise 1C: Define three variables (VarA, VarB, VarC), and print the statement:
"Hello VarA, I am sorry I'm late. My VarB ate my VarC"

Exercise 1D: Print only the positive part of the sentence below:
VarD = "He's lazy, he's ugly, but he smells good"
Tip: See "Indexing" and "Slicing" on page 7 & 8 in the handout

Exercise 2: Strings, Integers, and floats

Exercise 2A: Determine the data type of the following values using Spyder:
"One", 1, "1", 1.0

Exercise 2B: Use one of the integrated Python functions to convert the integer **3** into a float

Exercise 2C: Determine the data type of **True** and **False**?

Exercise 2D: Test out the following statements in Spyder and note if they are defined as **True** or **False**:
1 + 1 == 2
2 + 4 < 5
1 == 1.0
2 != 2

Exercise 3: Lists

Exercise 3A: Create a list called *burger_list containing* five ingredients that belong in a burger, along with five of your favorite numbers (floats or integers). Finally print the list.
Tip: "1" is not an integer

Exercise 3B: Use indexing to print out the third, seventh, and tenth item from *burger_list*.

Exercise 4: If-statements in Python

Exercise 4A: Write an if-statement that checks whether a number of your choice is greater than 68.

Tip: Remember indentation!

Exercise 4B: Write an *if statements*, that tests if the third, seventh and tenth element in *burger_list* is a string.

Tip: Use indexing

Exercise 5: For-loops in Python

Exercise 5A: Create a list of cute animals and use a for-loop to print each animal in the list

Tip: Remember indentation

Exercise 5B: Use a for-loop with an if-statement to print each string in burger_list with the message: "X really goes in a burger!"

Tip: type(X) == str:

Exercise 6: Open and write out a file

Exercise 6A: Open and read through the script called **get_fasta_header.py** inside the scripts directory in Spyder and note what you expect to happen once running it.

Exercise 6B: Add comments inside the script at all locations where you see a #. Talk with your partner about the following:

- What does the "w" stand for on line 4?
- Why is *header* assigned as `line[1:]`, and not just `line`, on line 10?
- Why is the *fasta* file not closed?

Exercise 6C: Run the script from the command line. Is the outcome as you expected?

Exercise 7: Biopython

Exercise 7A: Make sure you're using the correct environment. Open spyder and go to "preferences" → "python interpreter" and make sure the specified path is:

/home/gebt/micromamba/2025_envs/BTG_biopython/bin/python3
(It is an environment containing Python and the Python package Biopython)

Exercise 7B: Open and read through the script called **get_fasta_length.py** from the scripts/ directory in Spyder and note what you expect to happen when executing it. Run the script from the command line and observe the output.

Exercise 7C: In Spyder, add # at the beginning of lines 13 and 14. Then, talk to your partner about what you expect to happen when running the script now. Finally, run the script from the command line and observe the output.

Exercise 7D: Replace line 9 with the following:

```
fasta_file = sys.argv[1] #Takes the first argument as input
```

Run the script from the command line and use P_aeruginosa_TOprJ3-positive.fasta as argument for the script.

Tip: see system arguments in handout

Tip: Remember to activate the conda environment in the terminal:

conda activate BTG_biopython

Extra exercises

Extra Exercises 1:

Extra Exercises 1: While Loop

Exercise E1A: Create a variable called **sheep** and set it to 0. Using a while loop, print the number of **sheep** and increase the amount of **sheep** by 1 until you reach 23.

Exercise E1B: Copy and paste the duck list below into Spyder:

```
duck_list=["duck", "duck", "duck", "duck", "duck", "duck", "duck", "goose", "duck", "duck", "duck"]
```

Exercise E1C: Using a *for-loop* to iterate through duck_list, count the number of **ducks** and stop when you reach **goose**. Print "Goose!" along with the number of ducks counted before the goose.

Exercise E1D: Repeat exercise b, but use a *while-loop* instead of a for loop.
Tip: use len()

Extra Exercises 2: Making a function

Exercise E2A: Modify the previous if-statement from Exercise 4A by turning it into a function called higher_than(X). The function should take an input number X and return one of the following messages:

"X is lower than 68"
"X is higher than 68"
"both numbers are 68"

```
# Example result from Exercise 4A:
num = 70 # You can change this number to test different values

# If statement to check if the number is greater than 68
if num > 68:
    print(num, "is greater than 68")
```

Exercise E2B: Improve the function and make it take two inputs (integer or float) so that running higher_than(X,Y) will return one of the following:

"X is not higher than Y"
"X is higher than Y"
"Both numbers are X" *(if X and Y are the same)*

Extra Exercises 4: Dictionaries

Exercise E4A: Create a dictionary called *favorite_foods* with your top three favorite foods as keys and their corresponding deliciousness levels (on a scale of 1-10) as values. Print the dictionary to see your favorite foods and how delicious they are.

Exercise E4B: Choose one food and print its deliciousness level along with an explanatory message by retrieving the information from the dictionary.
Example: The deliciousness level of broccoli is 10

Exercise E4C: Modify the deliciousness level to make it even more delicious and make a new explanatory message.
Example: Actually, the deliciousness level of broccoli is more like 11

Exercise E4D: Add your least favorite food to the dictionary along with its corresponding deliciousness levels and update the explanatory message.
Example: I don't care for pizza; I only find its deliciousness level to be 3

Exercise E4E: Make a dictionary called *my_dict* with a key called **number** and a key called **food** that each have an empty list as value.

Exercise E4F: Make a *for-loop* that iterates through the *burger_list* created in Exercise 3A and append all string values (burger ingredients) to the **food**-key and all numerical values to the **number-**key.

```
# Example:
burger_list = ["lettuce", "cheese", "tomato", "bacon", "onion", 7, 3.14, 42, 10, 99.9]
```