Assignment 3 - Working with text

In this assignment you will familiarize yourself more with text in Python. Text is incredibly important because it is so ubiquitous. And it is nice because it requires a lot less math than coding graphics. Try to solve as much as you can, but do hand in. That is, hand in regardless of how much you have done. Do not forget to review the feedback you get!

Exercise A: What are these programs doing?

For this exercise you do not have to type in the code or run it. You can of course, but try **not** to.

The goal is for you to explain what the individual pieces of code below will print. Imagine someone runs the code; what do you think will be the output? Write one answer per program. Save your answers in `A.txt` .

In [ ]:

```python
alphabet = 'abcdefghijklmnopqrstuvxyzæøå'
print(alphabet[7] + alphabet[20] + alphabet[11] + alphabet[11] + alphabet[20]
```

In [ ]:

```python
c = 10
while c > 1:
    print(c)
    c -= 10
```

In [ ]:

```python
my_list = [1, 2, 3, 4]
for number in my_list:
    print(number)
```

Exercise B: Good coding practices

In the session we had an exercise on drawing using a while loop. Can you change the below code to:

- use better naming practices and
- use an augmented assignment operator like +=?

Save your answer in `B.py`.

```
from turtle import forward, left


a = 170
f = 200
c = 36
while c > 0:
    forward(f)
    left(a)
    c = c - 1
```

Exercise C: Working with lists

Say you have the following list stored in the variable `my_list`:

```
my_list = [4, 2, 8, 1, 9, -1, 8]
```

The program below is supposed to find the smallest number in the list, but it is not working as expected.

Your task is to:

- Enter the program in Mu, set a breakpoint, run the debugger, write down in text what is happening.
- Fix the program so it actually finds the smallest number

Save your answer in `C.py`

```
my_list = [4, 2, 8, 1, 9, -1, 8]

smallest_number = 1000000
for number in my_list:
    if number > smallest_number:
        smallest_number = number
print('I found the smallest number: ' + smallest_number)
```

Exercise D: Generating names

For this exercise you will need to download the file `us_names.py` and put it into the same directory that you are currently working in (it is likely the directory `mu_code` in your home directory). The file needs to be stored there as you need to import and use it as in the following:

In [ ]:

```
from us_names import SURNAMES, FEMALE_NAMES, MALE_NAMES
```

Translated from Python to English, that simply means "load the Python file `us_names.py` and make the three variables from within that file: `SURNAMES` , `FEMALE_NAMES` and `MALE_NAMES` accessible to my program". You can now use those variables directly in your code. Try to do that now.

The actual exercise is:

- Write a program that creates four valid US American names and prints them.
  - Of these four names are two female and two male names.
  - A valid name has the form `<name> <surname>` (with a space between the name and surname), such as, *Shayla Maeda* or *Hipolito Crytser*.

Save your answer in `D.py` .

Hints:

- The names are sorted so that the most common names are first and the rarest names are in the end.
- To create a name you need to lookup the name (gendered name + surname) in your variables
- Does it make sense to use loops in your program?

Exercise E: Randomly generated names

- The code below prints out a random number between 0 and 10. Can you use that to randomize your 4 names? That is, you are right now using fixed indexes to lookup the names in the variables, so they will always stay the same. Can you exploit the code below to create names that changes every time you execute the code?
  - Hint: You can find the length of a list, such as `SURNAMES` using the `len` function:

```
len(SURNAMES)
```

Save your answer in `E.py`

In [ ]:

```
from random import randint


a_random_number = randint(0, 10)
print(a_random_number)
```