Introduction to Python
Python I

1. Getting started

This is a short recap what you have learned the previous days.

1.1 Connecting to UPPMAX

Open a terminal and start a ssh connection to UPPMAX as you learned on Monday.

```
$ ssh -X <username>@rackham.uppmax.uu.se
```

Note: `<username>` is replaced by your own UPPMAX username.

1.2 Create new directory

Before you start with the following assignments, please create a directory `PythonLab1` in your home directory (`~/`). In this directory save all the files you create during this assignment.

1.3 Exercises

For the first assignment you should get familiar with the Python-interpreter. In the second assignment you write your first Python program using an editor. More instructions are given in the following how to use both, the Python-interpreter and an editor. Good luck!

2. Basic exercises

Load the module for Python 3 with the command `module load python3/3.6.0` Open the Python-interpreter with the command `python3` You should then see at the beginning of the line: `>>>.` In this exercise we use only the Python-interpreter. You can leave the Python-interpreter when you type `quit()`

2.1 Type in the Python-interpreter the following command:

```
print("Assignment")
```

What happens?

2.2 Enter now `i = 10` in the Python-interpreter
and then (in a new line) `print(i)`
After that (in a new line) enter `j = i/2`
and (in a new line) `print(j)`

Which values are displayed and why?

Hint: With `type()` the type of a variable can be determined. For example, `type("hello")` returns `<class 'str'>` which means that `"hello"` is of type string.

2.3 Assign to variable `7Assignment` the string **hurr durr**. Don't forget
to put the string in quotation marks (`"  "`).
Which error occurs and why?

2.4 Assign to variable `A` the sequence **AGCTA** (don't forget to put the sequence in quotation marks). Use the built-in function `len()` to determine the length of the sequence `A` and assign the length of `A` to variable `i`. Print `A` and `i`.

2.5 Concatenate `A` and `i` and print the result.
What happens and why?

2.6 Enter now `print(A + str(i))`
What happens now and why?

Hint: What might the built-in function `str()` do? There are also other built-in functions, e.g., to convert a string or number to an in- teger: `int()` or to convert a string or number to a floating point: `float()`

2.7 Print the substring of `A` from position 2 to 4.
The output should be: **GCT**

2.8 Print the prefix (beginning of a string) of length 2 and the suffix (end
of a string) of length 2 of the sequence stored in `A`. The output should be **AG** and **TA**.

2.9 Write a for-loop with the loop variable `i`, which runs from 0 to `len(A)`
and prints out `i`.

Hint: Don’t forget to indent the body of the for-loop.

Execute the same for-loop a second time and print out the character at each position of string `A` using `A[i]` as well.

2.10 Add now an if-condition inside the for-loop, which checks if `i < len(A)/2`
Only print i and `A[i]` if this condition is true.

2.11 Write a while-loop, which produces the same output as the for-loop
and if-condition together.

2.12 Print the variable `A` again. What happens?

2.13 Leave the interactive mode of Python with `quit()`

2.14 Now start Python interactively again and print the variable `A`.
What happens now and why?

3. First small program

Open your favorite editor (nano, gedit, etc.) and write in the file named `compare.py` your first Python program.

Hint: When you type

```
$ gedit compare.py&
```

in the terminal, a new line in the terminal should appear (if not press <ctrl C>). Then you can run your program in the same terminal window:

```
$ python3 compare.py
```

The advantage is that you can edit your program and switch easily between the editor and terminal window.

1. Write a short program which compares two variables i and j. It should print the value 1, if i and j are equal, and otherwise the value 0.

2. Within the program assign different numbers to i and j, e.g.:

a) `i = 3` and `j = 4`
and
b) `i = 10` and `j = 10`

Does your program work?

Congratulations, you have now completed the basic python exercises for this session. If you were too quick or just want to try a bit harder exercises, please continue with the bonus exercises below.

4. Bonus exercises

4.1 Sequences

In this exercise we write a short Python program (named `<program_name>.py`, think of a reasonable program name and name your file accordingly. Replace `<program_name>` with your new program name).

Chose two variables, e.g. `A` and `B` and assign the sequences **GATTACA** and **TACCATAC** to these variables. Make sure that the two sequences are assigned as strings to their variables `A` and `B`. Then print these sequences. Save everything you wrote and close the editor. Then you can run your program: `python3 <program_name>.py`

Then extend your program:

1) Concatenate both sequences in both ways (`AB` and `BA`) and print both options.

2) Print prefixes and suffixes of length 3 of both sequences `A` and `B`. Use the built-in function `len()` for determining the suffixes.

3) Print out the second sequence from the last to the first position (last

position first, first position last).

4) Assign this inverted sequence to a third variable, you could use the variable name `C`, and print the value of this variable.

5) Print out the middle base of each sequence. When a sequence has an even number of bases, print out the base at the right position of the middle. Use the built-in function `len()` for this task.

For example: For `A = "GTCA"` the program should print out `C`.

Hint: There exist built-in functions to convert a number to an integer.

6) Count how often each base occurs in the first sequence (How often does **G** occur in the first sequence, then **A**, so on.) and print out this number for each base.

7) Count how often does **TA** occur in the second sequence and print out this number.

4.2 Calculate the product of two numbers

Write in an editor the program `product.py` as introduced in the lecture, which calculates the product of two numbers **456** and **15**. Save the program code as `product.py` and run the program as described in the previous assignment.

1) Now calculate the product of **234** and **24** additionally to the first product and print out both products (results) in one single line.

2) Change the program so that all numbers **456**, **15**, **234**, and **24** are saved in one list, called `prod_list`. Change the print statement so that each number gets printed and also the product of the first two and the last two numbers.

4.3 More sequences

Write in an editor a program, which has three lists `l`, `m`, and `n`. Each list will contain several sequences described below. Save and run the program as described previously.

`l`: **AGGTC, GATC, CTGCA, ATTCGT, ATGGT, GATC**
`m`: **CTGCA, GATC**
`n`: **CUAGCUA, GTATGG, GUAUC, GTAG**

Note: Remember to store all sequences as strings in each of the lists.

Extend your program so that it can perform the following tasks:

1) Print each sequence in list `l`.

2) Print the first and last sequence in list `l`.

3) For each sequence in list `l` store the second position of each sequence in a new variable and print this new sequence.

4) Add this new sequence to the list `l`.

5) How long is list `l` now? Print out the length of list `l`.

6) Delete the second sequence of list `l`. Print list `l` and its length.

7) Divide the new list `l` into two equal parts and store the first half in a new list `l1` and the second half in a new list `l2`. Print both lists.

8) Concatenate list `l2` and list `l1` (in this order) and store it in a new list `l3`.

9) Remove all sequences in list `l`, which are also present in list `m`.

10) Invert all sequences in list `l` and store them in a new list `l4`.

11) In list `n` a few RNA sequences (**U** instead of **T**) are present.
- Change these sequences back to DNA sequences. - Delete the RNA sequences in list `n`. - Add the new DNA sequences at the same position of list `n`.
When you print list `n` it should contain the following sequences in this order: **CTAGCTA**, **GTATGG**, **GTATC**, and **GTAG**.