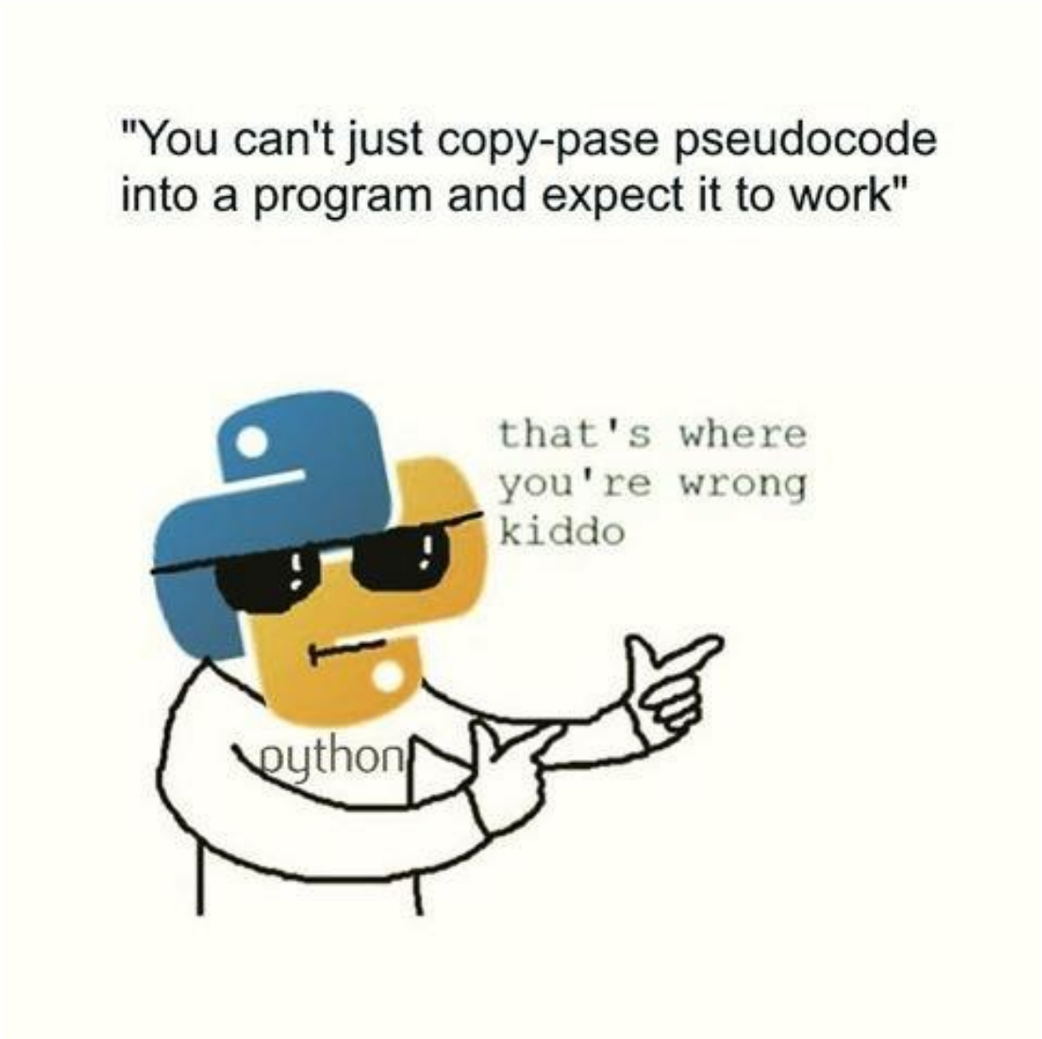
pythonclub 02



Start your VM, open your terminal (`Ctrl+Alt+T`) *(something else for Macs :P)*, then type in `python3`

pythonclub 02

Start your VM, open your terminal (`Ctrl+Alt+T`) and try to type the commands shown on screen, we're going to go through:

- lists
- dictionaries
- scripting
- if statements
- for loops

You will then work on a few exercises.
After what, we will explore `import` and a few advanced string manipulations.
The last exercise is Caesar's cipher, for which the solution is provided.

lists (1/2)

Compound datatypes often referred to as sequences, most common is `list`.

```
# empty list
>>> my_list = []
>>> my_list
# list of integers
>>> my_list = [1, 2, 3]
>>> my_list
# list with mixed datatypes
>>> my_list = [1, "Hello", 3.4]
>>> my_list
```

Get element of a list (first element is `0`):

```
>>> my_list[0] #This is one of the main differences between matlab and python
>>> my_list[1]
```

lists (2/2)

Negative indexing:

```
>>> my_list = ['a','b','c','d','e']
>>> my_list[-1]
>>> my_list[-2]
```

Add element to list:

```
>>> my_list = ['a','b','c','d','e']
>>> my_list.append('f')
>>> my_list
```

Modify list

```
# mistake values
>>> my_list = ['a','k','c','d','e']
# change the 1st item
>>> my_list[1] = "b"
>>> my_list
```

dicts (1/2)

An item has a key and the corresponding value expressed as a pair, `key: value`.

- Tip1:For people transitioning from Matlab to python, dicts are like structure in Matlab.
- Tip2:Transfering data from Matlab .mat to python can be done with dicts !

```
# empty dictionary
>>> my_dict = {}
>>> my_dict
# dictionary with integer keys
>>> my_dict = {1: 'pla', 2: 'peek'}
>>> my_dict
# get element of dict using key
>>> my_dict[1]

# dict with string keys
>>> my_dict = {'fdm': 'makerbot', 'sla': 'formlabs'}
>>> my_dict['sla']
```

dicts (2/2)

Update/add items to dicts:

```
>>> my_dict = {'fdm': 'makerbot', 'sla': 'formlabs'}

# update value
>>> my_dict['fdm'] = 'leapfrog'
>>> my_dict

# add item
>>> my_dict['solvencast'] = 'fisnar'
>>> my_dict
```

Get length of a list or dictionary:

```
>>> my_list = ['a','b','c','d','e']
>>> len(my_list)
>>> my_dict = {'fdm': 'makerbot', 'sla': 'formlabs'}
>>> len(my_dict)
```

Scripting

Open a terminal, you should be in your `home` directory, the prompt shoud look like this:

`ilyass@tx1~$`

- `ilyass` is the user,
- `tx1` is the name of the machine,
- `~` means that we are in our home folder,
- `$` means that you are a normal user

Let's create a folder:

`ilyass@tx1~$ mkdir Exercises` *(mkdir: make directory)*

Check the folder was created:

`ilyass@tx1~$ ls` *(ls: shows file/folders in current directory)*

Enter the folder:

`ilyass@tx1~$ cd Exercies` *(cd: change directory)*

Scripting

Notice how the terminal prompt changed to reflect the director we are in:

`ilyass@tx1~/Exercises$`

We are now going to write a python script in this folder. We will be using `atom`.

First, let's get `atom`. We can install software using the `apt-get` command:

`ilyass@tx1~/Exercises$ sudo snap install atom --classic`

- `sudo` means that we need admin privileges to execute this command
- `snap`: simple installation and update management (like an appstore, `apt-get` is the most popular one)
- `install` we want `snap` to use the `install` function
- `atom` it is the software we want to install
- `--classic` is because `atom` install uses the classic version of snap

Scripting

Now that we have `atom`, simply write:

`ilyass@tx1~/Exercises$ atom .`

Use `New File` to create an empty file, write a few words in it and save it as `temp.py`

Paste the following content in the file:

```
print("Hello world !")
```

Save, then go back to the terminal and input:

```
ilyass@tx1~/Exercises$ python3 temp.py
```

The file will now execute and should print `Hello world !`

> Now, go to https://github.com/pythonclubmtl/learning_python3, and take a few minutes to read `001b-space-tabs.md`.

if/else

```
if test expression:
    Something happens
elif test expression:
    Something else happens
else:
    Another something else happens
```

Only one of the `something` will be triggered. If conditons overlap, first one will overtake.

> Now, go to https://github.com/pythonclubmtl/learning_python3, copy the content of `ex_if.py` in one of your file and execute it using python3.

for loop

```
for val in sequence:
        Something happens
```

> Now, go to https://github.com/pythonclubmtl/learning_python3, copy the content of `ex_for.py` in one of your file and execute it using python3.

Exercices

1. Fibonacci

Write a script that asks the user how many Fibonacci numbers to generate, then generates them and prints them as a list. Fibonacci sequence: $x_n = x_{n-1} + x_{n-2}$

2. Divisors

https://www.practicepython.org/exercise/2014/02/26/04-divisors.html

3. Common elements in lists

```
l1 = ["42", "11", "33", "97", "63", "86", "4", "46", "72", "88", "59", "55", "13'
l2 = ["24", "98", "56", "59", "3", "42", "14", "37", "75", "5", "34", "63", "4" ]
```

Write a script which: Prints all elements common to both list, then prints all elements common to both list and divisible by 7 (see `ex_listcommon.py` for solution).

> Next time: functions, sets, read/write CSV

import

`import <module_name>`

Let's go ahead and open python3 (`Ctrl+Alt+T` input `python3` then `Enter`):

```
>>> import time
```

We just loaded everything included in the `time` package. Ond of the function is `sleep` .

```
>>> time.sleep(7)
```

Loading a package takes resources (memory, time). Sometimes we just want one function:

```
>>> from time import sleep
>>> sleep(2)
```

- Basic packages are provided with python (https://docs.python.org/3/library/), anyone can write a package that we can the download online.

Caesar's cipher: string package (1/2)

Need the alphabet as a string ?

```
>>> import string
>>> string.ascii_lowercase
```

Position of letter `t` in the alphabet (start counting from 0) ?

```
>>> import string
>>> alphabet = string.ascii_lowercase
>>> alphabet.find("t")
```

https://docs.python.org/3/library/string.html

Caesar's cipher: string package (2/2)

Add character to a new string using character's position in the alphabet:

```
>>> import string
>>> alphabet = string.ascii_lowercase
>>> new_character = alphabet[8]
>>> encrypted_message = ""
>>> encrypted_message += new_character
>>> new_character = alphabet[22]
>>> encrypted_message += new_character
>>> encrypted_message
```

You should now be able to write a nice Caesar's cipher encoder/decoder (solution available: `ex_caesarcipher.py`).