Day 01 – Piscine Python for Data Science

Intro to Python: Syntax and Semantics

Summary: This day will help you to get the basic knowledge about syntax and semantics of Python.

Contents

I.	Preamble	2
II.	Instructions	3
III.	Specific instructions of the day	4
IV.	Exercise 00	5
V.	Exercise 01	6
VI.	Exercise 02	7
VII.	Exercise 03	8
VIII.	Exercise 04	9
IX.	Exercise 05	10
X.	Exercise 06	11
XI.	Exercise 07	12

Chapter I

Preamble

Python is the most popular programming language for data science. Why is it so good for that kind of task? Python is an interpreted language. It means that you can easily interact with different pieces of code and get fast results. And that is exactly what we need if we want to analyze data from different angles or try different hyperparameters for a machine learning model. Besides this, Python has a lot of libraries that are suitable for scientific tasks including data science. Add to this a pretty simple syntax and you will get the most popular programming language for data science tasks.

Just for fun, look at these 19 beautiful guiding principles that influenced the design of Python:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one-- and preferably only one --obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea -- let's do more of those!

In case you forget any of them, you can just write in Python `import this` and you will get them shortly.

Chapter II

Instructions

- Use this page as the only reference. Do not listen to any rumors and speculations about how to prepare your solution.
- Here and further we use `Python 3` as the only correct version of Python.
- The python files for python exercises (d01, d02, d03) must have a block in the end: `if __name__ == '__main__'`.
- Pay attention to the permissions of your files and directories.
- To be assessed your solution must be in your GIT repository.
- Your solutions will be evaluated by your piscine mates.
- You should not leave in your directory any other file than those explicitly specified by the exercise instructions. It is recommended that you modify your .gitignore to avoid accidents.
- When you need to get precise output in your programs, it is forbidden to display a precalculated output instead of performing the exercise correctly.
- Have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.
- Your reference manual: mates / Internet / Google.
- Remember to discuss on the Intra Piscine forum.
- Read the examples carefully. They may require things that are not otherwise specified in the subject.
- And may the Force be with you!

Chapter III

Specific instructions of the day

- No code in the global scope. Use functions!
- Each file must be ended by a function call in a condition similar to:

```
if __name__ == '__main__':
    your_function( whatever, parameter, is, required )
```

- It is tolerated to place an error handling in this same condition.
- No imports allowed, except those explicitly mentioned in the section “Authorized functions” of the title block of each exercise.
- The exceptions raised by the `open()` function are not to be handled.
- The interpreter to use is `Python 3`.

Chapter IV

Exercise 00

Exercise 00 : Data types
Directory to store your solution : `ex00/`
Files to be in the directory : `data_types.py`
Authorized functions : `n/a`
Comments : `n/a`

Python as any other language has several built-in data types. In this exercise, you will get familiar with the most popular and useful ones.

Create a script called data_types.py in which you need to define a data_type() function. In this function, you need to declare 8 variables with different types and print them out on standard output. You must reproduce exactly the following output:

```
$ python3 data_types.py
[int, str, float, bool, list, dict, tuple, set]
```

It is forbidden to explicitly write the data types in your print. Remember to call your function at the end of your script as explained in the day instructions:

```
if __name__ == '__main__':
    data_types()
```

Put your file in the `ex00` folder in the root of your repository.

Chapter V

Exercise 01

Exercise 01 : Working with files
Directory to store your solution : `ex01/`
Files to be in the directory : `read_and_write.py`
Authorized functions : `n/a`
Comments : `n/a`

For this exercise, you are free to define as many functions as you need and to name them whatever you want. In the attached file `ds.csv` (you will recognize it from the previous day experience), you will have several columns separated by a comma with different data about vacancies.

Design a Python script called `read_and_write.py` whose role is to open the file `ds.csv`, read the data it contains, replace all the comma delimiters with '`\t`' and save it to another file `ds.tsv`.

Put your script in the `ex01` folder in the root of your repository.

Chapter VI

Exercise 02

Exercise 02 : Dictionaries
Directory to store your solution : `ex02/`
Files to be in the directory : `to_dictionary.py`
Authorized functions : `n/a`
Comments : `n/a`

Create a script named to_dictionary.py where you need to copy in one of your functions the list of the following tuples as is:

```
list_of_tuples = [
                ("Russia", "25"),
                ("France", "132"),
                ("Germany", "132"),
                ("Spain", "178"),
                ("Italy", "162"),
                ("Portugal", "17"),
                ("Finland", "3"),
                ("Hungary", "2"),
                ("The Netherlands", "28"),
                ("The USA", "610"),
                ("The United Kingdom", "95"),
                ("China", "83"),
                ("Iran", "76"),
                ("Turkey", "65"),
                ("Belgium", "34"),
                ("Canada", "28"),
                ("Switzerland", "26"),
                ("Brazil", "25"),
                ("Austria", "14"),
                ("Israel", "12")
]
```

Your script should transform this variable into a dictionary where the number is the key and a country is the value. It must display the dictionary on standard output accordingly to this precise formatting:

```
"25" : "Russia"
"132" : "France"
"132" : "Germany"
"178" : "Spain"
...
Think about why the order is not necessarily identical to the example.
```

Chapter VII

Exercise 03

Exercise 03 : Search by key
Directory to store your solution : `ex03/`
Files to be in the directory : `stock_prices.py`
Authorized functions : `import sys`
Comments : `n/a`

You have the following dictionaries to copy to one of your functions:

```
companies = {
    “Apple” : “AAPL”,
    “Microsoft” : “MSFT”,
    “Netflix” : “NFLX”,
    “Tesla” : “TSLA”,
    “Nokia” : “NOK”
 }

stocks = {
         “AAPL” : 287.73,
         “MSFT” : 173.79,
         “NFLX” : 416.90,
         “TSLA” : 724.88,
         “NOK” : 3.37
}
```

Write a program that takes as argument a name of a company (ex: Apple) and displays on the standard output the stock price (ex: 287.73). If you give the program an argument a company that is not from the dictionary, your script should display “Unknown company”. If there are no arguments or too many arguments, your program should do nothing and quit.

```
$ python3 stock_prices.py Tesla
724.88

$ python3 stock_prices.py Facebook
Unknown company

$ python3 stock_prices.py Tesla Apple
```

Chapter VIII

Exercise 04

Exercise 04 : Search by value and by key
Directory to store your solution : `ex04/`
Files to be in the directory : `ticker_symbols.py`
Authorized functions : `import sys`
Comments : `n/a`

You have the same two dictionaries from the previous exercise. You should copy them again in one of your functions of the script.

Create a program this time that takes a ticker symbol (ex: AAPL) and displays the company name and the stock price with space as the delimiter. The rest of the behavior must be identical to the previous exercise.

```
$ python3 ticker_symbols.py TSLA
Tesla 724.88

$ python3 stock_prices.py FB
Unknown ticker

$ python3 stock_prices.py TSLA AAPL
```

Chapter IX

Exercise 05

Exercise 05 : Search by value or by key
Directory to store your solution : `ex05/`
Files to be in the directory : `all_stocks.py`
Authorized functions : `import sys`
Comments : `n/a`

You still have those two dictionaries. And you still should copy them in one of your functions in the script.

Write a program that has the following behavior:

- the program must take as argument a string containing many expressions to find whatever you want, separated by a comma,
- for each expression of the string, the program must detect if it is a company name or the ticker symbol, or neither,
- the program should not be case-sensitive and be able to work with white spaces,
- if there are no arguments or too many arguments, the program displays nothing,
- when there are two commas in a row in the string, the program does not display anything,
- the program must display the results separated by a line break and use the following formatting:

```
$ python3 all_stocks.py “TSLA , apple, Facebook”
TSLA is a ticker symbol for Tesla
Apple stock price is 287.73
Facebook is an unknown company or an unknown ticker symbol

$ python3 all_stocks.py “TSLA,, apple”


$ python3 all_stocks.py TSLA AAPL
```

Chapter X

Exercise 06

Exercise 06 : Sorting a dictionary
Directory to store your solution : `ex06/`
Files to be in the directory : `dict_sorter.py`
Authorized functions : `n/a`
Comments : `n/a`

In this exercise, you have the dictionary from the `ex02` with the countries and numbers. You should copy it in one of your functions in the script.

Write a program that displays the country names sorted by descending numbers, then in alphabetical order of names if the numbers are equal. You need to display them one per line and without the numbers:

```
The USA
Spain
Italy
France
Germany
...
```

Chapter XI

Exercise 07

Exercise 07 : Caesar cipher
Directory to store your solution : `ex07/`
Files to be in the directory : `encoder.py, decoder.py`
Authorized functions : `import os, import sys`
Comments : `n/a`

There is such a thing as Caesar cipher that helps encode some text using a shift in the alphabet order. For example, the encoded version of "`hello`" might be "`tqxxa`" if we use the shift equal to `12`.

Write two programs: the first, `encoder.py`, will encode any string using a given shift, the second, `decoder.py`, will decode any string using a given shift.

```
$ python3 encoder.py "hello" 12
"tqxxa"

$ python3 decoder.py "tqxxa" 12
"hello"
```

If the scripts are given a string with, for example, Cyrillic symbols, the scripts should raise the exception "`The script does not support your language yet.`" If the scripts were given a string without quotes, raise an exception.