Basic Programming - Seminar 6

Alexandra Ciobotaru

14 Ianuarie 2021

This is a recap seminar.

1 Variables

Remember data types from Lesson 2?

Text Type:	str	x = "Hello World"
Numeric Types:	int	x = 20
	float	x = 20.5
	complex	x = 1j
Sequence Types:	list	x = ["apple", "banana", "cherry"]
	tuple	x = ("apple", "banana", "cherry")
	range	x = range(6)
Mapping Type:	dict	x = {"name" : "John", "age" : 36}
Set Types:	set	x = {"apple", "banana", "cherry"}
	frozenset	x = frozenset({"apple", "banana", "cherry"})
Boolean Type:	bool	x = True

Figure 1: Python data types

1.1 Create variables

Create a variable called `name` and assign to it your name. Next, print it in console.

1.2 Create a variable to be asisgned after input

Create a variable x that takes the input from user. Write a message inside the input function to be prompted to the user in Console:

```
x = input(...)
```

1.3 Variable type

What is the `type` of the variable x?

1.4 Create empty variables

Create empty variables of different data types (some of which could be used as accumulators later on) and print them using string formatting:

1. a zero integer
2. an empty string
3. a False Boolean
4. an empty list

1.5 Naming variables

Variables cannot be named:

1. starting with a number
2. using a reserved word: 'class', 'continue', 'in', 'is', 'list' etc.
3. using special charaters: % & etc.

1.6 Create a float

Create a float variable by doing a simple operation of division. Put the remainder of the division into another variable using the modulo (%) operator. What type is the remainder?

2 Lists

Remember that lists are defined using square brackets.

```
my_list = ['a', 'b', 'c']
```

2.1 Lists are ordered and dynamic

Meaning that the items in a list have a defined order, and that order will not change. If you add new items to a list, the new items will be placed at the end of the list, maintaining the initial order.

Exercise: add letter 'd' to `my_list` using method `append`. Verify your list by printing it. Next, pop it out using method `pop`, and verify your list again.

2.2 Lists can contain any arbitrary objects

Create a list of integers and assign it to a variable. Create another list, but containing strings, just like my_list, but containing words instead of characters.

Next, create a list of lists.

Let's go even further and create a list that contains a number, a character, and another list!

2.3 List elements can be accessed by index

In Python, indexing starts at 0 :)

Figure 2: Yet another Python indexing meme

Ok, on a more serious note, let's remember what indexing means by studying Figure 3.

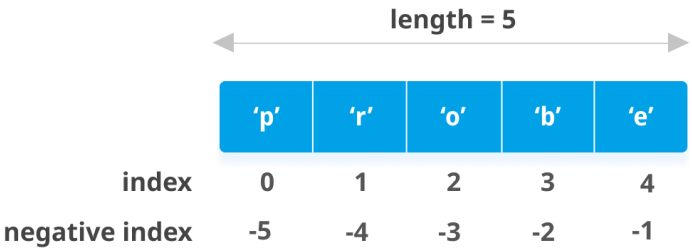

Figure 3: Python indexing

Suppose you have this list: `a = ['foo', 'bar', 'baz', 'qux', 'quux', 'corge']`

1. Print the element at index 0 in `a`. Print the last element of list `a`.
2. **String concatenation**: create a variable called `my_sum` and assign to it the sum of the first and the last element in list `a`.

3. Write a slice that would select `['bar','baz']` in list `a`.
4. Remember how you could reverse a list? Try typing `a[::-1]`.

2.4 Lists can be nested to arbitrary depth

Suppose you have the list: `x = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', ['hh', 'ii'], 'j']`. How can you acces, using list indexing, the element `'ccc'`?

2.5 Lists are mutable

Meaning that once a list has been created, elements can be added, deleted, shifted, and moved around at will.

Exercise: delete element in `a` using the `del` keyword. Revwrite the initial list and try to remove the same element using method `remove`.

2.6 Tuples

Create a tuple containing the same elements as list `a`. Remember tuples are immutable (they cannot be changed): try to assign a different value to `a[2]`. Use method `len` to get the length of your tuple. Note that `len` is a built-in function that can compute the length of various datatypes.

3 Strings

You can define a string variable in multiple ways:

1. x = 'Hello'
2. x = "Hello" (this is useful for strings inside strings)
3. x = """Hello there""" (useful for writing on multiple lines)

3.1 String indexing

Strings can be indexed just like lists. Suppose you have the string `my_string = "Hello there human"`.

3.2 String slicing

Remember that `my_string[m:n]` returns the portion of `my_string` starting with position m, and up to but not including position n.

1. Write a slice that selects the word 'human' and then assign it to a variable.
2. Write a slice that selects everything in that string up until index 5.

3. write a slice that selects everything in string from index 5 up until the end of that string.

3.3 Strings are considered immutable

Meaning you cannot change a value in string directly, you need to use a method: `my_string = my_string.replace('b', 'x')` Using slicing and concatenation, replace in `my_string` the word 'there' with 'beautiful'.

3.4 Common methods

1. index() - gets the index of a specific character in text;
2. join - having a list of strings it concatenates all elements in it: `combined_string = " ".join(["1", "2", "3"]);`
3. split() - does the opposite, splits text by space or something specified in method;
4. strip() - eliminates spaces or other characters from the beginning and the end of a string;
5. and more... upper, lower, capitalize, startswith, endswith, isdigit, isalpha etc.

3.5 String formatting

```
print("I love %s in %s" % ("programming", "Python"))
```

```
print("I love programming in python".format(programming="programming",
python="Python"))
```

3.6 Exercise

Write a short line of code to test if character 'i' is in string 'mine'.

4 Iterators

4.1 The FOR loop

1. Write a for loop where you write a letter in 'panseluta' for each interation.
2. Write a for loop where you iterate and print `i` in a `range(5)`.
3. Create a list accumulator named `acc`, and, using a for loop, iterate through `values = [1, 2, 3, 4]` and append to the accumulator the values in `values`, squared.

4.2 The WHILE loop

Create a while loop that loops forever, and while doing that it takes the input from keyboard and rewrites it.

HINT:

```
while True:
    print("something... ", input())
```

4.3 Conditionals

Write a for loop in which you'll count how many times character 'a' appears in word 'panseluta'. Use an integer accumulator and the += formula to accumulate. HINT:

```
acc = ...
    for char in ...:
        if ... == 'a':
            ...
```

Next, create an else branch where you print "this character is not a", and the character that is not a.

Remember this pseudocode?

Write some if/elif statements for the conditions date == 1 and date == 2.

Note: using ifs instead of elifs will execute all lines of code with a True condition.

4.4 Comparison and Logical Operators

Please go through `https://www.w3schools.com/python/python_operators.asp` to remember operators and their meaning.

Write an expression that tests is variable a is less or equal than variable b, and if True, it prints which variable "is bigger", else it prints which variable "is lesser".

5 Functions

Let's remember the syntax of python functions in Figure 5.

5.1 Docstrings

A function can also have a docstring - which is a description of what that function does, what arguments need to be passed into it and what it returns (see Figure 6). Write a function called `my_prod` that returns the product of two variables, a and b. Explain in the docstring what your function does and read the docstring as in Figure 6.

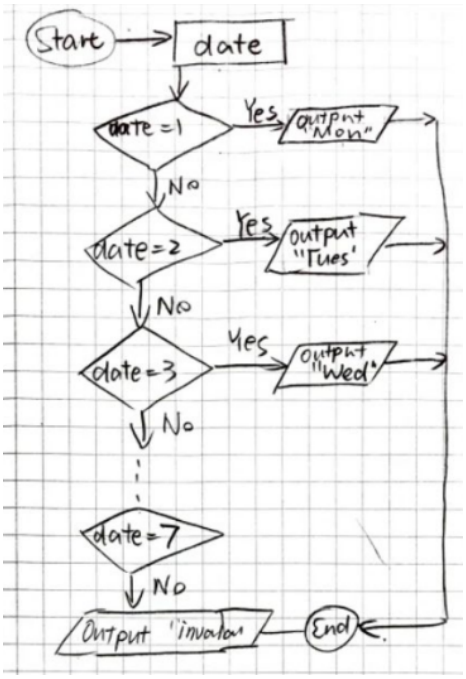

Figure 4: If/else conditionals

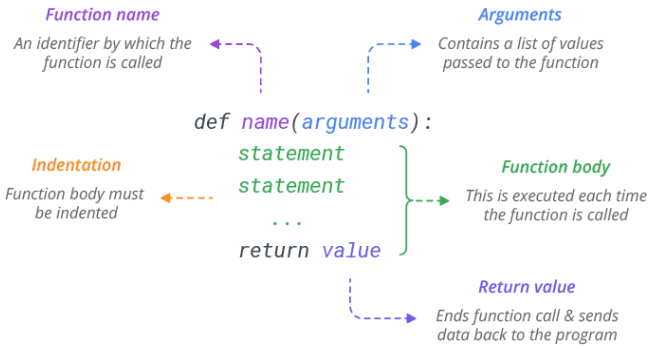

Figure 5: Python function explained.

```
def cat():
    """This function returns the string meow when called"""
    return "meow"

cat() # "meow"

cat.__doc__ # "This function returns the string meow when called"

help(cat) # gives us even more detail with the docstring!
```

Figure 6: Python docstring

5.2 Exercise

Create a function named `count_even`. It should accept one input argument, named 'numbers', which will be a list containing integers. Your function should return the number of even integers in the input list. Don't forget to write it's docstring - this is important so you will later remember what your function does!

6 Libraries

6.1 Installing and using libraries

Installing libraries is done using the command `pip install library_name` in the Terminal. Next, in order to be used, that library should be imported at the beginning of your code: `import library_name`. **Exercise** Using `langdetect` library and it's method `detect`, write a function that detects the language from input keyboard and returns a string in the following format: `"you wrote something in ", language`.

7 Vectors and Matrices

Create a function that takes as input 2 integer variables, x and y, and returns a matrix of ones with x rows and y lines.

HINT: use `np.ones((x,y))`.

Write some lines of code to add two vectors:

```
a = [1,2,3,4]
b = [3,4,5,6]
```

HINT: use `np.array(a)` to create an array out of list a, do the same with b, then add them using built-in function `sum`.

8 Sets and Dictionaries

Given the set: `my_set = {'apples', 'bananas', 'pears'}` Add some keys to it and transform it into a dictionary. Remember dictionaries are key-value pairs. Write a for loop to print it's keys. Can you modify the loop to print its values?

Add to your dictionary the key-value pair: `'hello':True` and delete the first key.

Given the second set `my_second_set = {'cherries', 'mango', 'pears'}`, use `union` method to compute the reunion of the two. Compute the intersection using `intersection`.

Cast your two sets into lists and create a dictionary using zip, like in this example:

```
number_list = [1, 2, 3]
str_list = ['one','two','three']
result = zip(number_list, str_list)
print(dict(result))
```

9 JSON

Having the following JSON:

```
import json
studentJson ="""{
    "id": 1,
    "name": "john wick",
    "class": 8,
    "percentage": 75,
    "email": "jhon@pynative.com"
}"""
```

Check if 'percentage' key exists in studentJSON. First, you need to load it into a JSON object.

1. if you load a JSON from a .json file - use json.load()
2. if you load a JSON from string - use json.loads()

Save this example JSON into a .json file and load it into an object (using `load` method. Modify the value at percentage and dump it into another file. Further, dump the same object into a JSON string using `.loads` method.

10 File handling

Create a text file and open it.

```
f = open("demofile.txt", "r")
for x in f:
    print(x)
f.close()
```

The with open statement is easier to use as it automatically closes the document as well.

```
with open("demofile.txt", "r") as f:
    for x in f:
        print(x)
```

Remember opening modes are 'r' for reading, 'a' for appending without modifying the previous existing lines and 'w' for writing.

Loop through the file line by line and add to a counter the number of lines that files has.

11 Working with dataframes

To work with pandas, you first need to import the pandas library as pd (that's standard), and install it if you didn't (see section 6). Next, open a csv at will and save it into a dataframe:
`df = pd.read_csv('my_data.csv')`

11.1 View your dataframe

Print the dataframe shape, using `shape`. View all it's columns names using:

```
for col in df.columns:
    print(col)
```

Drop rows containing missing values using `dropna` method. Select a column in dataframe using `df["column_name"]` and print in console all its elements.

11.2 Selecting elements

Try `loc` and iloc to select some elements in your dataframe.

```
loc -> selects [row_label, column_label]
iloc -> selects [row_position, column_position]
```

11.3 Save dataframe to csv

Create a new dataframe by slicing the first 10 rows and write it to a csv using `to_csv` metghod.

12 OOP

Write a Python program to create a Vehicle class with `max_speed` and `mileage` instance attributes.

```
class ...:
    def __init__(self, ..., ...):
        self.max_speed = max_speed
        self.mileage = mileage
modelX = Vehicle(240, 18)
print(modelX.max_speed, modelX.mileage)
```

Congratulations, you have finished Basic Programming in Python!