Lab -2

Introduction to Python and computer programming

Instructor : AALWAHAB DHULFIQAR

Advisor : Dr. Tejfel Mate

<u>What you will learn:</u>

How to get Python and how to get to use it

Starting your work with Python

How to write and run your very first program

How to write and run your very first program

How to spoil and fix your code

how to write and run simple Python programs;

how to perform basic input and output operations.

PYTHON INSTITUTE

Open Education & Development Group

How to write and run simple Python programs;

```
print("Hello, World!")
```

The `print()` function

Where do the functions come from?

As you can see, the first program consists of the following parts:

1. the word print;
2. an opening parenthesis;
3. a quotation mark;
4. a line of text: Hello, World!;
5. another quotation mark;
6. a closing parenthesis.

1. They may come from Python itself; the print function is one of this kind; such a function is an added value received together with Python and its environment (it is built-in); you don't have to do anything special (e.g., ask anyone for anything) if you want to make use of it;
2. They may come from one or more of Python's add-ons named modules; some of the modules come with Python, others may require separate installation - whatever the case, they all need to be explicitly connected with your code (we'll show you how to do that soon);
3. you can write them yourself, placing as many functions as you want and need inside your program to make it simpler, clearer and more elegant.

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

Note:
a function may have:
- an effect;
- a result.

The `print()` function

```
print("Hello, World!")
```

function invocation

function_name(argument)

1. First, Python checks if the name specified is legal (it browses its internal data in order to find an existing function of the name; if this search fails, Python aborts the code);
2. second, Python checks if the function's requirements for the number of arguments allows you to invoke the function in this way (e.g., if a specific function demands exactly two arguments, any invocation delivering only one argument will be considered erroneous, and will abort the code's execution);
3. third, Python leaves your code for a moment and jumps into the function you want to invoke; of course, it takes your argument(s) too and passes it/them to the function;
4. fourth, the function executes its code, causes the desired effect (if any), evaluates the desired result(s) (if any) and finishes its task;
5. finally, Python returns to your code (to the place just after the invocation) and resumes its execution.

When we encounter any new function, we must answer these three questions:

1- **What is the effect of the function**

2- What arguments needed

3-What value does the function return

Lab2.1

You have already seen a computer program that contains one function invocation. A function invocation is one of many possible kinds of Python instructions.

Of course, any complex program usually contains many more instructions than one. The question is: how do you couple more than one instruction into the Python code?

Python's syntax is quite specific in this area. Unlike most programming languages, Python requires that there cannot be more than one instruction in a line.

A line can be empty (i.e., it may contain no instruction at all) but it must not contain two, three or more instructions. This is strictly prohibited.

Note: Python makes one exception to this rule - it allows one instruction to spread across more than one line (which may be helpful when your code contains complex constructions).

This is a good opportunity to make some observations:

the program invokes the print() function twice, and you can see two separate lines in the console - this means that print() begins its output from a new line each time it starts its execution; you can change this behavior, but you can also use it to your advantage;

each print() invocation contains a different string, as its argument and the console content reflects it - this means that the instructions in the code are executed in the same order in which they have been placed in the source file; no next instruction is executed until the previous one is completed (there are some exceptions to this rule, but you can ignore them for now)

```
print("The itsy bitsy spider climbed up the waterspout.")
print("Down came the rain and washed the spider out.")
```

Expected output :

```
The itsy bitsy spider climbed up the waterspout.
Down came the rain and washed the spider out.
```

Lab2.1 Cont.

```
print("The itsy bitsy spider climbed up the waterspout.")
print()
print("Down came the rain and washed the spider out.")
```

```
print("The itsy bitsy spider\nclimbed up the waterspout.")
print()
print("Down came the rain\nand washed the spider out.")
```

The backslash (\) has a very special meaning when used inside strings - this is called the escape character.

```
print("\")
```

The `print()` function - using multiple arguments

```
print("The itsy bitsy spider" , "climbed up" , "the waterspout.")
```

The `print()` function - the keyword arguments

Keyword	Example

end	print("My name is", "Python.", end=" ")
	print("Monty Python.")

sep	print("My", "name", "is", "Monty", "Python.", sep="-")

```
print("My", "name", "is", sep="_", end="*")
print("Monty", "Python.", sep="*", end="*\n")
```

Lab 2.2

Objectives

experimenting with existing Python code;
discovering and fixing basic syntax errors;
becoming familiar with the print() function and its formatting capabilities.

Scenario

We strongly encourage you to play with the code we've written for you, and make some (maybe even destructive) amendments. Feel free to modify any part of the code, but there is one condition - learn from your mistakes and draw your own conclusions.

Try to:

1. minimize the number of print() function invocations by inserting the \n sequence into the strings
2. make the arrow twice as large (but keep the proportion)
3. duplicate the arrow, placing both arrows side by side; note: a string may be multiplied by using the following trick: "string" * 2 will produce "stringstring" (we'll tell you more about it soon)
4. remove any of the quotes, and look carefully at Python's response; pay attention to where Python sees an error - is this the place where the error really exists?
5. do the same with some of the parentheses;
6. change any of the print words into something else, differing only in case (e.g., Print) - what happens now?
7. replace some of the quotes with apostrophes; watch what happens carefully.

```
print("    *")
print("   * *")
print("  *   *")
print(" *     *")
print("***   ***")
print("  *   *")
print("  *   *")
print("  *****")
```

Expected output :

Solve the 7 points after Try to. For any questions, feel free to ask me in the consultation hours.

Lab 2.2

Print the following

"I'm"

""learning""

"""Python""“

Copy past the following to your IDLE, What is the output

```
print(0.000000000000000000001)

print(0o123)

print(0x123)

print("2")
print(2)
```

$\left.\right\}$ What is the difference here ?

Use your brain as IDLE

What is the type of the following

"Hello ", "007“

"1.5", 2.0, 528, False

Basic operators - **Arithmetic operators**

operators	Python example

exponentiation (power)	print(2 ** 3) print(2 ** 3.)
	print(2. ** 3) print(2. ** 3.)

multiplication	print(2 * 3) print(2 * 3.)
	print(2. * 3) print(2. * 3.)

division	print(6 / 3) print(6 / 3.)
	print(6. / 3) print(6. / 3.)

integer division	print(-6 // 3) print(6 // 3.)
	print(6. // 3) print(6. // 3.)

remainder (modulo)	print(14 % 4) print(12 % 4.5)
addition	print(-4 + 4) print(-4. + 8)

subtraction	print(-4 - 4) print(4. - 8)
	print(-1.1)

Do not try to:

- perform a division by zero;
- perform an integer division by zero;
- find a remainder of a division by zero.

Operators and their priorities   2 + 3 * 5 = ?

Operators and their bindings   print(9 % 6 % 2) = ?   print(2 ** 2 ** 3) = ?

List of priorities

Priority	Operator	
1	+, -	unary
2	**	
3	*, /, //, %	
4	+, -	binary

Solving simple mathematical problems

$c = \sqrt{a^2 + b^2}$

```
a = 3.0
b = 4.0
c = (a ** 2 + b ** 2) ** 0.5
print("c =", c)
```

Use your brain as IDLE
What is the type of the following

What is the output ?

1. print(2 * 3 % 5)
2. print((5 * ((25 % 13) + 100) / (2 * 13)) // 2)
3. print((2 ** 4), (2 * 4.), (2 * 4))
4. print((-2 / 4), (2 / 4), (2 // 4), (-2 // 4))
5. print((2 % -4), (2 % 4), (2 ** 3 ** 2))

Lab 2.3 (related to variables, slides 9-11 from the lecture part)

Objectives

becoming familiar with the concept of storing and working with different data types in Python;
experimenting with Python code.

Scenario

Here is a short story:

Once upon a time in Appleland, John had three apples, Mary had five apples, and Adam had six apples. They were all very happy and lived for a long time. End of story.

Your task is to:

- create the variables: john, mary, and adam;
- assign values to the variables. The values must be equal to the numbers of fruit possessed by John, Mary, and Adam respectively;
- having stored the numbers in the variables, print the variables on one line, and separate each of them with a comma;
- now create a new variable named total_apples equal to addition of the three former variables.
- print the value stored in total_apples to the console;
- experiment with your code: create new variables, assign different values to them, and perform various arithmetic operations on them (e.g., +, -, *, /, //, etc.). Try to print a string and an integer together on one line, e.g., "Total number of apples:" and total_apples.

Lab 2.4 (related to variables, slides 9-11 from the lecture part)

Objectives

- becoming familiar with the concept of, and working with, variables;
- performing basic computations and conversions;
- experimenting with Python code.

Scenario

Miles and kilometers are units of length or distance.

Bearing in mind that 1 mile is equal to approximately 1.61 kilometers, complete the program in the editor so that it converts:

- miles to kilometers;
- kilometers to miles.

Do not change anything in the existing code. Write your code in the places indicated by ###. Test your program with the data we've provided in the source code.

Pay particular attention to what is going on inside the print() function. Analyze how we provide multiple arguments to the function, and how we output the expected data.

Note that some of the arguments inside the print() function are strings (e.g., "miles is", whereas some other are variables (e.g., miles).

```
kilometers = 12.25
miles = 7.38

miles_to_kilometers = ###
kilometers_to_miles = ###

print(miles, "miles is", round(miles_to_kilometers, 2), "kilometers")
print(kilometers, "kilometers is", round(kilometers_to_miles, 2), "miles")
```

Expected output :

```
7.38 miles is 11.88 kilometers
12.25 kilometers is 7.61 miles
```

Lab 2.5 (see slide 9-10 from this file)

Objectives

becoming familiar with the concept of numbers, operators, and arithmetic operations in Python;
performing basic calculations.

Scenario

Take a look at the code in the editor: it reads a float value, puts it into a variable named x, and prints the value of a variable named y. Your task is to complete the code in order to evaluate the following expression:

3x3 - 2x2 + 3x - 1

The result should be assigned to y.

Remember that classical algebraic notation likes to omit the multiplication operator - you need to use it explicitly. Note how we change data type to make sure that x is of type float.

Keep your code clean and readable, and test it using the data we've provided, each time assigning it to the x variable (by hardcoding it). Don't be discouraged by any initial failures. Be persistent and inquisitive.

```
x =  # hardcode your test data here
x = float(x)
# write your code here
print("y =", y)
```

Sample input
x = 0
x = 1
x = -1

Expected Output
y = -1.0
y = 3.0
y = -9.0

Lab 2.6 (related to variables, slides 12 from the lecture part)

Objectives

- becoming familiar with the concept of comments in Python;
- using and not using comments;
- replacing comments with code;
- experimenting with Python code.

Scenario

The code in the editor contains comments. Try to improve it: add or remove comments where you find it appropriate (yes, sometimes removing a comment can make the code more readable), and change variable names where you think this will improve code comprehension.

```
#this program computes the number of seconds in a given number of hours
# this program has been written two days ago

a = 2 # number of hours
seconds = 3600 # number of seconds in 1 hour

print("Hours: ", a) #printing the number of hours
# print("Seconds in Hours: ", a * seconds) # printing the number of seconds
in a given number of hours

#here we should also print "Goodbye", but a programmer didn't have time
to write any code
#this is the end of the program that computes the number of seconds in 3
hour
```

Use your brain as IDLE

What is the type of the following

What is the output ?

```
var = 2
var = 3
print(var)
```

```
a = '1'
b = "1"
print(a + b)
```

```
a = 6
b = 3
a /= 2 * b
print(a)
```

```
my_var
m
101
averylongvariablen
ame
m101
m 101
Del
del
```

```
# print("String #1")
print("String #2")
```

```
# This is a multiline
comment. #
print("Hello!")
```

Lab 2.7 (related to variables, slides 13 from the lecture part)

Objectives

- improving the ability to use numbers, operators, and arithmetic operations in Python;
- using the print() function's formatting capabilities;
- learning to express everyday-life phenomena in terms of programming language.

Scenario

Your task is to prepare a simple code able to evaluate the end time of a period of time, given as a number of minutes (it could be arbitrarily large). The start time is given as a pair of hours (0..23) and minutes (0..59). The result has to be printed to the console.

For example, if an event starts at 12:17 and lasts 59 minutes, it will end at 13:16.

Don't worry about any imperfections in your code - it's okay if it accepts an invalid time - the most important thing is that the code produce valid results for valid input data.

Test your code carefully. Hint: using the % operator may be the key to success.

```
hour = int(input("Starting time (hours): "))
mins = int(input("Starting time (minutes): "))
dura = int(input("Event duration (minutes): "))

# Write your code here.
```

Sample input:
12
17
59

Expected output: 13:16

Congratulations

You have learned :

- the basic methods of formatting and outputting data offered by Python, together with the primary kinds of data and numerical operators, their mutual relations and bindings;
- the concept of variables and variable naming conventions;
- the assignment operator, the rules governing the building of expressions;
- the inputting and converting of data;

See you in the Next week ☺