Lab-02: Python programming (Syntax, printing, data types and variables, conditional loops)

2.1 Objectives:

- To get familiar with the syntax of Python
- To get familiar with data types and variables in Python
- To get familiar with conditional loops in Python
- To get familiar with the print function in Python

2.2 Learning python for a C++/C# programmer

Let us try to quickly compare the syntax of python with that of C++/C#:

	C++/C#	**Python**
Comment begins with	//	#
Statement ends with	;	No semi-colon needed
Blocks of code	Defined by {}	Defined by indentation (usually four spaces)
Indentation of code and use of white space	Is irrelevant	Must be same for same block of code (for example for a set of statements to be executed after a particular if statement)
Conditional statement	if-else if-else	if – elif – else:
Parentheses for loop execution condition	Required	Not required but loop condition followed by a colon : while a < n: print(a)

2.3 Math in Python

Calculations are simple with Python, and expression syntax is straightforward: the operators +, -, * and / work as expected; parentheses () can be used for grouping.

```
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3     #Exponent operator
8
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>> 17 // 3  # floor division
5
>>> 23%3       #Modulus operator
2
```

2.4 Python Operators

Command	Name	Example	Output
+	Addition	4+5	9
-	Subtraction	8-5	3
*	Multiplication	4*5	20
/	Classic Division	19/3	6.3333
%	Modulus	19%3	5
**	Exponent	2**4	16
//	Floor Division	19/3	6

2.5 Comments in Python:

```
#I am a comment. I can say whatever I want!
```

2.6 Variables:

```
print ("This program is a demo of variables")
```

```
v = 1

print ("The value of v is now", v)

v = v + 1

print ("v now equals itself plus one, making it worth", v)

print ("To make v five times bigger, you would have to type v = v * 5")

v = v * 5

print ("There you go, now v equals", v, "and not", v / 5 )
```

2.6.1 Strings:

```
word1 = "Good"

word2 = "Morning"

word3 = "to you too!"

print (word1, word2)

sentence = word1 + " " + word2 + " " +word3

print (sentence)
```

2.7 Relational operators:

Expression	Function
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
!=	not equal to
==	is equal to

2.8 Boolean Logic:

Boolean logic is used to make more complicated conditions for **if** statements that rely on more than one condition. Python's Boolean operators are **and**, **or**, and **not**. The **and** operator takes two arguments, and evaluates as **True** if, and only if, both of its arguments are True. Otherwise it evaluates to **False**. The **or** operator also takes two arguments. It evaluates if either (or both) of its arguments are **False**. Unlike the other operators we've seen so far, **not** only takes one argument and inverts it. The result of **not True** is **False**, and **not False** is **True**.

2.9 Operator Precedence:

Operator	Description
()	Parentheses
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus
* / % //	Multiply, divide, modulo, and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive 'OR' and regular 'OR'
<= < > >=	Comparison Operators
== !=	Equality Operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

2.10 Conditional Statements:

'if' - Statement

```
y = 1
if y == 1:
print ("y still equals 1, I was just checking")
```

'if - else' - Statement

```
a = 1
if a > 5:
    print ("This shouldn't happen.")
else:
    print ("This should happen.")
```

'elif' - Statement

```
z = 4
if z > 70:
    print ("Something is very wrong")
elif z < 7:
    print ("This is normal")
```

2.11 Input from user:

The **input()** function prompts for input and returns a string.

```
a = input ("Enter Value for variable a: ")
print (a)
```

2.12 The print() function

Three important questions have to be answered as soon as possible:

1. What is the effect the print() function causes?

The effect is very useful and very spectacular. The function:

- takes its arguments (it may accept more than one argument and may also accept less than one argument)
- converts them into human-readable form if needed (as you may suspect, strings don't require this action, as the string is already readable)
- and sends the resulting data to the output device (usually the console); in other words, anything you put into the print() function will appear on your screen.

No wonder then, that from now on, you'll utilize print() very intensively to see the results of your operations and evaluations.

2. What arguments does print() expect?

Any. We'll show you soon that print() is able to operate with virtually all types of data offered by Python. Strings, numbers, characters, logical values, objects - any of these may be successfully passed to print().

3. What value does the print() function evaluate?

None. Its effect is enough - print() does not evaluate anything.

2.13 The print() function - instructions

You already know that this program contains one function invocation. In turn, the function invocation is one of the possible kinds of Python instruction. Ergo, this program consists of just one instruction.

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).

2.14 TASKS:

2.14.1 TASK 1:

Write a program that first displays a simple cafe menu (see example below), asks the user to enter the number of a choice, and either prints the appropriate action OR prints an error message that their choice was not valid.

Example output:

1. Soup and salad
2. Pasta with meat sauce
3. Chef's special

Which number would you like to order? 2 One Pasta with meat sauce coming right up!

Another example output:

1. Soup and salad
2. Pasta with meat sauce
3. Chef's special

Which number would you like to order? 5

Sorry, that is not a valid choice.

2.14.2 TASK 2:

Once upon a time in Apple land, 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 totalApples equal to addition of the three former variables.

• print the value stored in totalApples to the console

• Check if the totalApples is greater, smaller or equal to 10