Assignment No. 1:Basics Of Python

Python is a general purpose, dynamic, high level and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn. It also provides high level data structures. The implementation of Python was started in the December 1989 by Guido Van Rossum at CWI in Netherland**.**

Starting the Interpreter

After installation, the python interpreter lives in the installed directory. By default, it is /usr/local/bin/pythonX.X in Linux/Unix and C:\PythonXX in Windows, where the 'X' denotes the version number. To invoke it from the shell or the command prompt we need to add this location in the search path.

Search path is a list of directories (locations) where the operating system searches for executables. For example, in the Windows command prompt, we can type set path=%path%; c:\python33 (python33 means version 3.3, it might be different in your case) to add the location to the path for that particular session.

Use of Python

Python is used by hundreds of thousands of programmers and is used in many places.

Python has many standard libraries which are made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

· Web development

· AI & Machine learning

· Game programming

· Desktop GUI Applications

· Software Development · Business Applications

· 3D CAD Applications

· Scientific programming

· Network programming.

First Python Program

This is a small example of a Python program. It shows "Hello World!" on the screen. Type the following code in any text editor or an IDE and save as *helloWorld.py print ("Hello, world!")*

Now at the command window, go to the location of this file. One can use the cd command to change directory. To run the script, type **python helloWorld.py** in the command window. We should be able to see the output as follows: **Hello, world!**

1. **Immediate/Interactive mode**

   Typing python in the command line will invoke the interpreter in immediate mode. We can directly type in Python expressions and press enter to get the output. >>>

   is the Python prompt. It tells us that the interpreter is ready for our input. Try typing in 1 + 1 and press enter. We get 2 as the output. This prompt can be used as a calculator. To exit this mode type exit () or quit () and press enter.
   Type the following text at the Python prompt and press the Enter –

   >>> print "Hello World"

2. **Script Mode Programming**

   Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

   Let us write a simple Python program in a script. Python files have extension

   .py. Type the following source code in a *test.py* file −

   ```
   print"Hello World"
   ```

   We assume that you have a Python interpreter set in the PATH variable.
   Now, try to run this program as follows −

   ```
   $ python
   test.py
   ```

   This produces the following result −

   ```
   Hello,
   Python!
   ```

   IDEL is a graphical user interface (GUI) that can be installed along with the

   Python programming language and is available from the official website.

Python Comments

In Python, there are two ways to annotate your code.

Single-line comment – Comments are created simply by beginning a line with the hash (#) character, and they are automatically terminated by the end of line.

For example:

```
#This would be a comment in
Python
```

Multi-line comment- Comments that span multiple lines – used to explain things in more detail – are created by adding a delimiter (‘‘‘’’’) on each end of the comment.

```
”””” This would be a multiline comment

In Python that spans several lines and

Describes your code, your day, or anything you want
it to

...
“””
```

Indentation: The enforcement of indentation in Python makes the code look neat and clean.

For example: if True:

print(‘Hello’)

a=5

Incorrect indentation will result in Indentation Error.

Standard Data Types

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers: Integers, floating point numbers and complex numbers fall under the Python numbers category. They are defined as int, float and complex classes in Python. Python supports four different numerical types –

- int (signed integers)
- long (long integers, they can also be represented in octal and hexadecimal)
- float (floating point real values)
- complex (complex numbers)

Python Strings: Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes.

Example: str=’Hello all’

Python Lists:

Lists are the most versatile of Python's compound data types. A list containing items can be of different data types separated by commas and enclosed within square brackets

([]).

```
list_obj=['table', 59 ,2.69,“chair”]
```

Python Tuples:

A tuple is another sequence immutable data type that is similar to the list. A

tuple consists of a number of values separated by commas and enclosed in parentheses (()).

```
Example:

tuple_obj=(786,2.23, “college”
)
```

Python Dictionary

Python's dictionaries are a kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs.

Dictionaries are enclosed by curly braces ({})

```
Example:dict_obj={'roll_no': 15,'name':’xyz’,'per': 69.88}
```

Python Operators:

Python language supports the following types of operators.

- Arithmetic Operators
- · Comparison (Relational) Operators
- Assignment Operators
- Logical Operators ● Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic, logical, Relational operators supported by Python language are same as other languages like C, C++.

i. Arithmetic Operators:

The new arithmetic operators in python are,

a) ** (Exponent)- Performs exponential (power) calculation on operators

Example: a**b =10 to the power 20

b) // (Floor Division) - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity)
Example: 9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

ii. Logical operators:

Logical operators are the and, or, not operators.

a) and - True if both the operands are true
b) or - True if either of the operands is true
c) not - True if operand is false (complements the operand)

iii. Relational/Comparison operators:

== (equal to), != (not equal to), < (less than),<= (**Less than or equal to**), > (greater than) and >= (**Greater than or equal to**) are same as other language relational operators. The new relational operator in python is, <>- If values of two operands are not equal, then condition becomes true.

Example: (a <> b) is true. This is similar to! = operator.

iv. Assignment Operators: The following are assignment operators in python which are same as in C, C++.

=, +=, -=, *=, /=, %=, **=, //=

v. Bitwise Operators: The following are bitwise operators in python which are same as in C, C++. & (bitwise AND), | (bitwise OR), ^ (bitwise XOR), ~ (bitwise NOT), << (bitwise left shift), >> (bitwise right shift)

vi. Membership operators: **in** and **not in** are the membership operators; used to test whether a value or variable is in a sequence.
in - True if value is found in the sequence not
in - True if value is not found in the sequence

vii. Identity operators: is and **is not** are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal

does not imply that they are identical. is - True if the operands are identical is not - True if the operands are not identical

Decision making Statement

Python programming language provides following types of decision making statements.

i. **If statement**:

It is similar to that of other languages

Syntax:

```
if expression:
statement(s)
```

 # python program to illustrate If

```
i = 10
if (i > 15):
    print("10 is less than 15")
print("I am Not in if")
```

Output-

I am Not in if

As the condition present in the if statement is false. So, the block below the if statement is not executed.

if-else

The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else if the condition is false. Here comes the *else* statement. We can use the *else* statement with *if* statement to execute a block of code when the condition is false.

Syntax:

```
if (condition):
    # Executes this block if
    # condition is true
else:
    # Executes this block if
```

```
   # condition is false
```

 # python program to illustrate if-else

```
age = 18

if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote yet.")
```

Output-
You are eligible to vote.

<u>if-elif-else</u>

Here, a user can decide among multiple options. The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.

Syntax:-

```
if condition1: # Code block executed if condition1 is True
elif condition2: # Code block executed if condition2 is True (if condition1 is False)
else: # Code block executed if all previous conditions are False
```

 # python program to illustrate if-elif-else

```
number = 15

if number > 20:
    print("The number is greater than 20.")
elif number == 15:
    print("The number is equal to 15.")
else:
     print("The number is less than 15.")
```

Nested-if

A **nested if** is an if statement that appears inside another if or else block. It allows you to check multiple conditions in a hierarchical manner, where one condition is dependent on the result of another.

```
if condition1:
                # Code block 1
    if condition2:
                # Code block 2 (this block is inside the first if)
                # Executes if both condition1 and condition2 are True
    else:
               # Code block 3 (this block executes if condition2 is False)
else:
                #Code block 4 (this block executes if condition1 is False)
```

```
age = 25
experience = 3
```

 # python program to illustrate nested-if

```
if age >= 18:
    print("You are old enough to apply for the job.")
    if experience >= 2:
        print("You have enough experience.")
 else:
        print("You do not have enough experience.")
else:
        print("You are too young to apply for the job.")
```

Python Loops

while loop:

A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

```
while condition:
    # Code to execute
```

 # python program to illustrate while loop

```
count = 1
while count <= 3:
    print("Hello")
    count += 1
```

for loop:

It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax

```
for variable in sequence:
        # Code to execute
```

 # python program to illustrate for loop

```
for num in range(1, 6):
    print(num)
```

Set A]

1) Write a Python Program to Calculate the Average of Numbers in a Given List.

2) Write a program which accepts 6 integer values and prints “DUPLICATES” if any of the values entered are duplicates otherwise it prints “ALL UNIQUE”. Example: Let 5 integers are (32, 10, 45, 90, 45, 6) then output “DUPLICATES” to be printed.

3) Write a program which accepts an integer value from user and print “Ok” if value is between 1 to 50 (both inclusive) otherwise it prints” Out of range”

4) Write a program which finds sum of digits of a number.
Example n=130 then output is 4 (1+3+0).

5) Write a program which prints Fibonacci series of a number.

Set B]

1) Write a program which accept an integer value ‘n’ and display all prime numbers till ‘n’.

2) Write a program that accept two integer values and if both are equal then prints “SAME identity” otherwise prints, “DIFFERENT identity”.

3) Write a program to display following pattern.

```
1 2 3 4
1 2 3
1 2
1
```

4) Write a program to reverse a given number.

Assignment Evaluation

0: Not Done [ ] 1 : Incomplete [ ] 2 : Late Complete[]

3: Needs Improvement [ ] 4 : Complete [ ] 5 : Well Done [ ]

Signture of Instructor