TCS332 Fundamental of Information Security and Blockchain

B. Tech CSE III Semester

Instructor:

Dr Mohammad Wazid

Professor, Department of CSE

Head of Cyber security and IoT research group

Graphic Era (Deemed to be University), Dehradun, India

Email: wazidkec2005@gmail.com

Homepage: https://sites.google.com/site/mwazidiiith/home

Introduction to Python

- Python is an object-oriented programming language created by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) (national research institute for mathematics and computer science), Netherlands in 1989.
- It is ideally designed for rapid prototyping of complex applications.
- It has interfaces to many OS system calls and libraries and is extensible to C or C++.
- Many large companies use the Python programming language include NASA, Google, BitTorrent, etc.

Introduction to Python

- Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks, Information Security and other advanced fields of Computer Science.
- Python had deep focus on code readability.

First python program

- **print("Welcome to GEU")**
- Type this in a notepad or gedit and save the file like **filename.py**
- Compilation and running.
- This will be done in one step. Just type following

python filename.py

- Then following output will come:

mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python welcome.py

Welcome to GEU

Python input and output:

Some of the functions like input() and print() are widely used for standard input and output operations respectively.

```
val = input("Enter your value: ")
print(val)
```

Addition of numbers

```
a = int(input("enter first number: "))
b = int(input("enter second number: "))
sum = a+b
print("sum:", sum)
```

Output:

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python addition.py
enter first number: 1
enter second number: 3
('sum:', 4)
```

Python Keywords

- Keywords are the reserved words in Python.
- We cannot use a keyword as a variable name, function name or any other identifier.
- They are used to define the syntax and structure of the Python language.
- In Python, keywords are case sensitive.
- There are 33 keywords in Python 3.7. This number can vary slightly in the course of time.
- Examples:

False, True, return, continue, try, if, in, elif, else, for, while, def, import.

Python Identifiers

An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.
Rules for writing identifiers.
Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _.
Examples:
myClass, var_1 and print_this_to_screen, all are valid example.

Python Statement

- Instructions that a Python interpreter can execute are called statements.
- For example, a=1 is an assignment statement.
- We could also put multiple statements in a single line using semicolons, as follows:

  a = 1; b = 2; c = 3

Python Indentation

- Most of the programming languages like C, C++, Java use braces { } to define a block of code. Python uses indentation.
- A code block (body of a function, loop etc.) starts with indentation and ends with the first unindented line.
- The amount of indentation is up to you, but it must be consistent throughout that block.
- Generally four whitespaces are used for indentation and is preferred over tabs.

Python Indentation: Example

```
for i in range(1,11):
    print(i)
    if i == 5:
        break
```

Output

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python
indentation.py
1
2
3
4
5
```

Python Comments

- Comments are very important while writing a program.
- It describes what's going on inside a program so that a person looking at the source code does not have a hard time figuring it out.
- In Python, we use the hash (#) symbol to start writing a comment.

```
#This is a comment
#print out Hello
print('Hello')
```

Python Variables

A variable is a named location used to store data in the memory.
It is helpful to think of variables as a container that holds data which can be changed later throughout programming.
For example,
number = 10

Data types in Python

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

There are various data types in Python. Some of the important types are listed below.

Python Numbers

Integers, floating point numbers and complex numbers falls under Python numbers category.

They are defined as int, float and complex class in Python.

Data types in Python
Python Numbers

- Integers, floating point numbers and complex numbers falls under Python numbers category.
- They are defined as int, float and complex class in Python.
- We can use the type() function to know which class a variable or a value belongs to and the isinstance() function to check if an object belongs to a particular class.

Data types in Python: Example

```
a = 5
print(a, "is of type", type(a))
a = 2.0
print(a, "is of type", type(a))
a = 1+2j
print(a, "is complex number?", isinstance(1+2j,complex))
```

Output:

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python
python-data-types.py
(5, 'is of type', <type 'int'>)
(2.0, 'is of type', <type 'float'>)
((1+2j), 'is complex number?', True)
```

Type Conversion

The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion. Python has two types of type conversion.

- Implicit Type Conversion
- Explicit Type Conversion

Implicit Type Conversion:

In Implicit type conversion, Python automatically converts one data type to another data type.
This process doesn't need any user involvement.

Type Conversion

Let’s see an example where Python promotes conversion of lower datatype (integer) to higher data type (float) to avoid data loss.

Implicit Type Conversion, Example 1: Converting integer to float

```
num_int = 123
num_flo = 1.23
num_new = num_int + num_flo
print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))
print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))
```

Type Conversion

Implicit Type Conversion,
Example 1: Converting integer to float

Output:

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python itc.py
('datatype of num_int:', <type 'int'>)
('datatype of num_flo:', <type 'float'>)
('Value of num_new:', 124.23)
('datatype of num_new:', <type 'float'>)
```

Type Conversion

- In the above program, we add two variables num_int and num_flo, storing the value in num_new.
- We will look at the data type of all three objects respectively.
- In the output we can see the datatype of num_int is an integer, datatype of num_flo is a float.
- Also, we can see the num_new has float data type because Python always converts smaller data type to larger data type to avoid the loss of data.

Type Conversion

Explicit Type Conversion:

- In Explicit Type Conversion, users convert the data type of an object to required data type.
- We use the predefined functions like int(), float(), str(), etc to perform explicit type conversion.
- This type conversion is also called typecasting because the user casts (change) the data type of the objects.

Syntax :

(required_datatype)(expression)

Type Conversion

Example 3: Addition of string and integer using explicit conversion

```
num_int = 123
num_str = "456"
print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type
Casting:",type(num_str))
num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))

num_sum = num_int + num_str
print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```

Type Conversion

Output:

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python etc.py
('Data type of num_int:', <type 'int'>)
('Data type of num_str before Type Casting:', <type 'str'>)
('Data type of num_str after Type Casting:', <type 'int'>)
('Sum of num_int and num_str:', 579)
('Data type of the sum:', <type 'int'>)
```

Python Operators: Arithmetic operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication etc.

Arithmetic operators in Python Operator Meaning Example

+ Add two operands or unary plus x + y

- Subtract right operand from the left or unary minus x - y

* Multiply two operands x * y

/ Divide left operand by the right one (always results into float) x / y

% Modulus - remainder of the division of left operand by the right x % y (remainder of x/y)

** Exponent - left operand raised to the power of right x**y (x to the power y)

Python Operators: Comparison operators

- Comparison operators are used to compare values. It either returns True or False according to the condition.

Comparison operators in Python	Operator	Meaning	Example
>	Greater than - True if left operand is greater than the right	x > y
<	Less than - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

Python Operators: Logical operators

- Logical operators are the and, or, not operators.

Logical operators in Python Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Python Operators: Bitwise operators

- Bitwise operators act on operands as if they were string of binary digits. It operates bit by bit, hence the name.
- In the table below: Let x = 10 (0000 1010 in binary) and y = 4 (0000 0100 in binary)

Bitwise operators in Python Operator	Meaning	Example
&	Bitwise AND	x& y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x>> 2 = 2 (0000 0010)
<<	Bitwise left shift	x<< 2 = 40 (0010 1000)

Python Operators: Assignment operators

- Assignment operators are used in Python to assign values to variables.
- a = 5 is a simple assignment operator that assigns the value 5 on the right to the variable a on the left.
- There are various compound operators in Python like a += 5 that adds to the variable and later assigns the same. It is equivalent to a = a + 5.

Python Operators: Assignment operators

Assignment operators in Python Operator	Example	Equivalent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5 (Floor Division.)
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Python Operators: Membership operators

- **In, not** in are the membership operators in Python.
- They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).
- In a dictionary we can only test for presence of key, not the value.

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

Python Operators: Conditional statements

If example:

```
# python program to illustrate If statement

i = 10
if (i > 15):
  print ("Correct")
print ("Incorrect")
```

Output:

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python if.py
Incorrect
```

Python Operators: Conditional statements

```
# python program to illustrate If else statement

i = 20
if (i < 15):
        print ("i is smaller than 15")
        print ("i'm in if Block")
else:
        print ("i is greater than 15")
        print ("i'm in else Block")
```

Output:

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$
python if-else.py
i is greater than 15
i'm in else Block
```

Python Operators: Conditional statements

Python program to illustrate if-elif-else ladder

```
print("Enter a number")
i = int(input())
if (i == 10):
        print ("i is 10")
elif (i == 15):
        print ("i is 15")
elif (i == 20):
        print ("i is 20")
else:
        print ("i is not present")
```

Python Operators: Conditional statements

 # Python program to illustrate if-elif-else ladder

Output:

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python if-else-elif.py
Enter a number
10
i is 10
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python if-else-elif.py
Enter a number
15
i is 15
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python if-else-elif.py
Enter a number
20
i is 20
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python if-else-elif.py
Enter a number
50
i is not present
```

Use of loops in python

While Loop:

- In python, while loop is used to execute a block of statements repeatedly until a given a condition is satisfied.
- And when the condition becomes false, the line immediately after the loop in program is executed.

Syntax :

```
while expression:
    statement(s)
```

Use of loops in python

```
# Python program to illustrate
# while loop
count = 0
while (count < 5):
        count = count + 1
        print("Welcome")
```

Output:

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python
while.py
Welcome
Welcome
Welcome
Welcome
Welcome
```

Use of loops in python

for loop:

- The for loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects.
- Iterating over a sequence is called traversal.

Syntax of for Loop

```
for val in sequence:
        Body of for
```

Use of loops in python

Program1:

```
for x in range(6):
  print(x)
```

Output:

```
0
1
2
3
4
5
```

Program1:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Output:

```
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python for-simple.py
apple
banana
cherry
```

Use of loops in python

```
# Program to find the sum of all numbers stored in a list
# List of numbers
numbers = [1, 2, 3, 4, 5]

# variable to store the sum
sum = 0
# iterate over the list
for val in numbers:
        sum = sum+val
# Output sum
print("The sum is", sum)
mwazid@mwazid:~/geu-ddn/fundamenals-cyber-sec/python$ python for-sum.py
('The sum is', 15)
```

References

Python for beginners book by Harsh Bhasin, 1st Edition 2019 , New Age International (P) Ltd.
Python: The Complete Reference book by Martin Brown and Martin C. Brown.