
ONE WEEK STTP ON “PYTHON PROGRAMMING”

DAY I – 22 JUNE 2020

by

R.CHITHRA DEVI

ASSOCIATE PROFESSOR

DEPARTMENT OF INFORMATION TECHNOLOGY

DR.SIVANTHI ADITANAR COLLEGE OF ENGINEERING

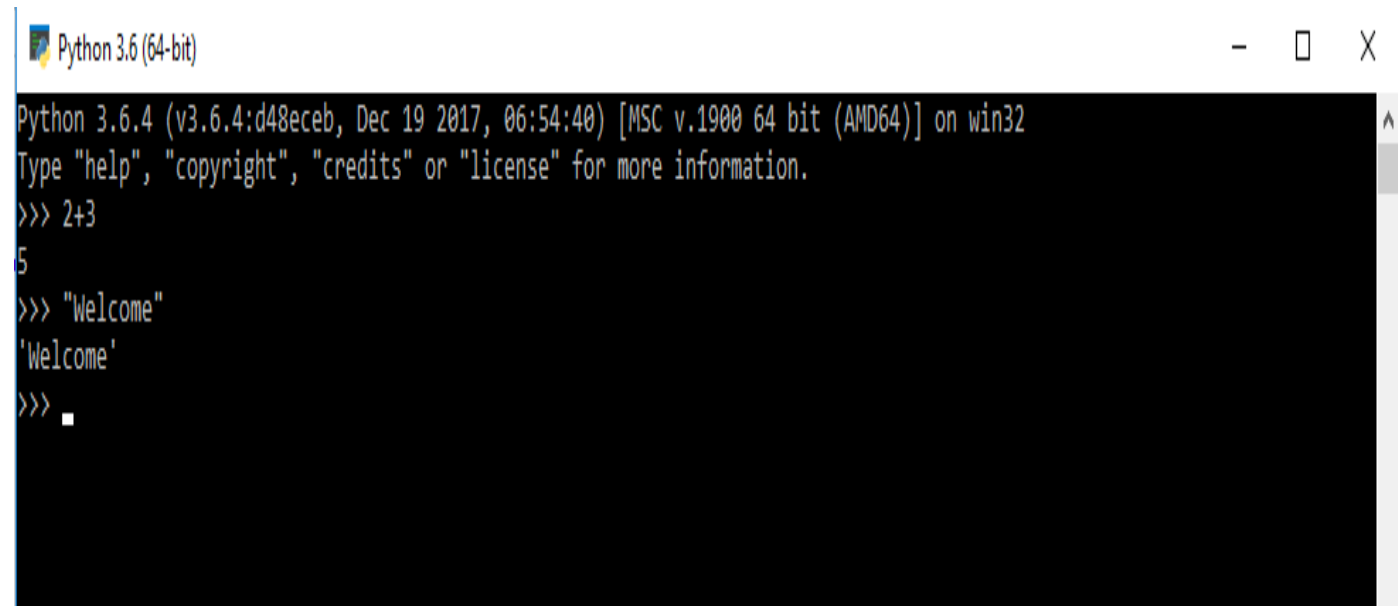
TIRUCHENDUR - 628205

FIRST PYTHON PROGRAM

- Execute the programs in different modes.
 - Interactive Mode Programming
 - Script Mode Programming

INTERACTIVE MODE PROGRAMMING

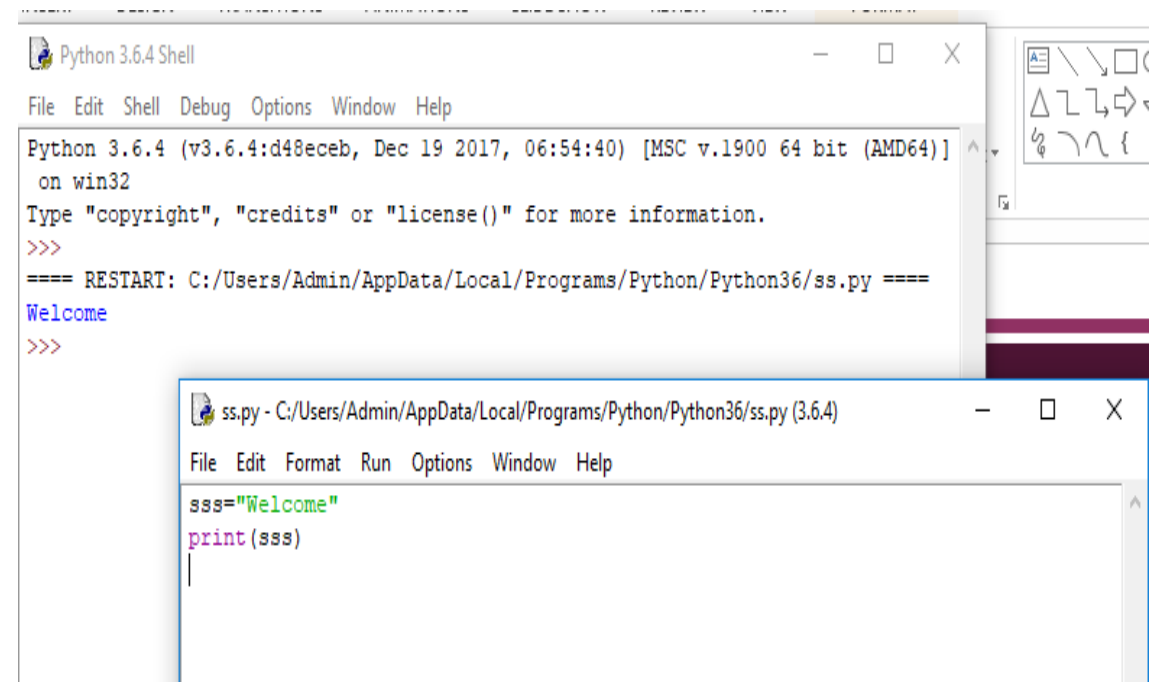
- Once Python is installed, typing python in the command line will invoke the interpreter in immediate mode.
- We can directly type in Python code, and press Enter to get the output.
- This prompt can be used as a calculator.
- To exit this mode, type quit() and press enter.
- For example,

A screenshot of a Windows command prompt window titled "Python 3.6 (64-bit)". The window shows the Python 3.6.4 interpreter running on Windows 32-bit. The prompt is ">>>". The user has entered "2+3" and the output is "5". The user has entered "Welcome" and the output is "'Welcome'". The user has entered a period "." and the output is ".".

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+3
5
>>> "Welcome"
'Welcome'
>>> .
.
```

SCRIPT MODE PROGRAMMING - IDE

- IDE - Integrated Development Environment
- When you install Python, an IDE named IDLE is also installed
- We can use any text editing software to write a Python script file.
- When you open IDLE, an interactive Python Shell is opened.
- Now you can create a new file and save it with .py extension.
- Write Python code in the file and save it.
- To run the file, go to Run > Run Module or simply click F5.
- For example,



PYTHON KEYWORDS

- Keywords are the reserved words in Python.
- We cannot use a keyword as a variable name, function name or any other identifier.
- In Python, keywords are case sensitive.

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

RULES FOR WRITING IDENTIFIERS

1. 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 `_`.
2. An identifier cannot start with a digit.
3. Keywords cannot be used as identifiers.
4. We cannot use special symbols in identifiers.
5. An identifier can be of any length.
6. Python is a case-sensitive language.
7. Multiple words can be separated using an underscore.

PYTHON STATEMENT

- Instructions that a Python interpreter can execute are called statements.

Multi-line statement

- The end of a statement is marked by a newline character (\).

```
a = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
```

- Multiple statements in a single line using semicolons.

```
a = 1; b = 2; c = 3
```

CONTINUATION...

- Python does not use braces({}) to indicate blocks of code for class and function definitions or flow control.
- Blocks of code are denoted by line indentation, which is rigidly enforced.
- For example –

```
if True:
```

```
    print ("True")
```

```
else:
```

```
    print ("False")
```


PYTHON COMMENTS

- We use the hash (#) symbol to start writing a comment.
- All characters after the #, up to the end of the physical line, are part of the comment and the Python interpreter ignores them.

First comment

print ("Welcome to STTP!") # second comment

This produces the following result –

Welcome to STTP!

QUOTATION IN PYTHON

- Python accepts single ('), double (") and triple (""" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.
- The triple quotes are used to span the string across multiple lines.
- For example,

```
word = 'word'
```

```
sentence = "This is a sentence."
```

```
paragraph = """This is a paragraph. It is  
made up of multiple lines and sentences."""
```

CLEAR WINDOWS

- `>>> import os`
- `>>> clear = os.system("cls")`

STANDARD DATA TYPES

- The data stored in memory can be of many types.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types –
 - Numbers
 - String
 - List
 - Tuple
 - Dictionary

PYTHON - NUMBERS

- Number data types store numeric values.
- Number objects are created when you assign a value to them.
- For example,

```
>>> abc=10
```

```
>>> xyz=150
```

```
>>> abc
```

```
10
```

```
>>> xyz
```

```
150
```

PYTHON - NUMBERS

- You can also delete the reference to a number object by using the **del** statement.
- You can delete a single object or multiple objects by using the **del** statement.
- **For example,**

```
>>> del abc
```

```
>>> abc
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
NameError: name 'abc' is not defined
```

```
>>> xyz
```

```
150
```

```
>>> aa=9
```

```
>>> bb=5
```

```
>>> aa
```

```
9
```

```
>>> bb
```

```
5
```

```
>>> del aa,bb
```

PYTHON - NUMBERS

- Python supports three different numerical types –
 - int (signed integers)
 - float (floating point real values)
 - complex (complex numbers)

For example,

int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	9.322e-36j

PYTHON - STRINGS

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows either pair of single or double quotes.
- Subsets of strings can be taken using the slice operator ([] and [:])
- Indexes starting at 0 in the beginning of the string and working their way from -1 to the end.

- For example,

```
>>> it="Welcome to Python STTP"
>>> it
'Welcome to Python STTP'
>>> print(it)
Welcome to Python STTP
>>> print(it[0])
W
>>> print(it[8:10])
to
>>> print(it[11:])
Python STTP
>>> print(it[-4:22])
STTP
```


PYTHON - STRINGS

- The plus (+) sign is the string concatenation operator.
- The asterisk (*) is the repetition operator.
- For example,

```
>>> print(it)
Welcome to Python STTP
>>> print("Hearty " + it)
Hearty Welcome to Python STTP
>>> print(it+" By IT Department")
Welcome to Python STTP By IT
Department
>>> print(it*2)
Welcome to Python STTPWelcome
to Python STTP
```

PYTHON - LISTS

- Lists are the most versatile of Python's compound data types.
 - List contains items separated by commas and enclosed within square brackets ([]).
 - Lists are similar to arrays in C.
 - One of the differences between them is that all the items belonging to a list can be of different data type.
 - The values stored in a list can be accessed using the slice operator ([] and [:])
 - Indexes starting at 0 in the beginning of the list and working their way to end -1.
 - The plus (+) sign is the list concatenation operator.
 - The asterisk (*) is the repetition operator.
- For example,

```
>>> list1 = [ "Tiruchendur", 1234 , 15.95, "IT", 100.5 ]
>>> list2 = [ "Dr.SACOE", "IT", "Department" ]
>>> print(list1)
['Tiruchendur', 1234, 15.95, 'IT', 100.5]
>>> print(list2)
['Dr.SACOE', 'IT', 'Department']
>>> print(list1+list2)
['Tiruchendur', 1234, 15.95, 'IT', 100.5, 'Dr.SACOE', 'IT', 'Department']
>>> print(list2*2)
['Dr.SACOE', 'IT', 'Department', 'Dr.SACOE', 'IT', 'Department']
>>> print(list1[1:4])
[1234, 15.95, 'IT']
>>> print(list1[3:])
['IT', 100.5]
```

PYTHON - TUPLES

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas.
- Unlike lists, however, tuples are enclosed within parenthesis.
- The values stored in a tuple can be accessed using the slice operator ([] and [:])
- Indexes starting at 0 in the beginning of the tuple and working their way to end -1.
- The plus (+) sign is the concatenation operator.
- The asterisk (*) is the repetition operator.

- For example,

```
>>> tuple1 = ( "Dr.SACOE", "IT", "Department" )
>>> tuple2 = ("Welcome", 1234)
>>> print(tuple1)
('Dr.SACOE', 'IT', 'Department')
>>> print(tuple2)
('Welcome', 1234)
>>> print(tuple1+tuple2)
('Dr.SACOE', 'IT', 'Department', 'Welcome', 1234)
>>> print(tuple2*3)
('Welcome', 1234, 'Welcome', 1234, 'Welcome', 1234)
>>> print(tuple1[0])
Dr.SACOE
>>> print(tuple1[1:3])
('IT', 'Department')
>>> print(tuple1[1:])
('IT', 'Department')
```

PYTHON - TUPLES

- The main difference between lists and tuples are –
- Lists are enclosed in brackets ([]) and their elements and size can be changed,
- Tuples are enclosed in parentheses (()) and cannot be updated.
- Tuples can be thought of as **read-only** lists.

- For example,

```
>>> list2=["Welcome",1234]
>>> tuple2=("Welcome",1234)
>>> print(list2)
['Welcome', 1234]
>>> print(tuple2)
('Welcome', 1234)
>>> list2[1]=5000
>>> print(list2)
['Welcome', 5000]
>>> tuple2[1]=5000
```

Traceback (most recent call last):

```
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not
support item assignment
```

PYTHON - DICTIONARY

- Python's dictionaries are kind of hash-table type.
- A dictionary consist of key-value pairs.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

- For example,

```
>>> dict1={}
>>> dict1['one'] = "ONE"
>>> dict1[2] = "TWO"
>>> print(dict1)
{'one': 'ONE', 2: 'TWO'}
>>> print(dict1['one'])
ONE
>>> print(dict1[2])
TWO
>>> dict2 = {'name': "STTP",'code':7, 'dept': "IT"}
>>> print(dict2)
{'name': 'STTP', 'code': 7, 'dept': 'IT'}
>>> print(dict2.keys())
dict_keys(['name', 'code', 'dept'])
>>> print(dict2.values())
dict_values(['STTP', 7, 'IT'])
```

TYPES OF OPERATOR

- Python language supports the following types of operators –
 - Arithmetic Operators
 - Comparison (Relational) Operators
 - Assignment Operators
 - Logical Operators
 - Bitwise Operators
 - Membership Operators
 - Identity Operators

PYTHON ARITHMETIC OPERATORS

Operator	Description	Example (a=10, b=21)
+ Addition	Adds values on either side of the operator	$a + b = 31$
- Subtraction	Subtracts right hand operand from left hand operand	$a - b = -11$
* Multiplication	Multiplies values on either side of the operator	$a * b = 210$
/ Division	Divides left hand operand by right hand operand	$b / a = 2.1$
% Modulus	Divides left hand operand by right hand operand and returns remainder	$b \% a = 1$
** Exponent	Performs exponential (power) calculation on operators	$a^{**}b = 10 \text{ to the power } 21 = 10000000000000000000000$
//	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)	$9//2 = 4$ and $9.0//2.0 = 4.0$, $-11//3 = -4$, $-11.0//3 = -4.0$

PYTHON COMPARISON OPERATORS

Operator	Description	Example (a=10, b=21)
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true = False
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true = True
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true = False
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true. = True
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true = False
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true = True

PYTHON ASSIGNMENT OPERATORS

Operator	Description	Example (a=10,b=21,c=1)
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a c /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a

PYTHON BITWISE OPERATORS

Operator	Description	Example a=60 (00111100), b=13 (00001101)
& Binary AND	Operator copies a bit, to the result, if it exists in both operands	(a & b) = 12 (means 0000 1100)
 Binary OR	It copies a bit, if it exists in either operand.	(a b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit, if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011)
<< Binary Left Shift	Operator shifts the left operand bits towards the left side. The left side bits are removed. Binary number 0 is appended to the end.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	Exactly opposite to the left shift operator. The left operand bits are moved towards the right side, the right side bits are removed. Binary number 0 is appended to the beginning.	a >> 2 = 15 (means 0000 1111) ₂₆

PYTHON LOGICAL OPERATORS

Operator	Description	Example a=1 (True), b=0 (False)
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is False. (means 0)
or Logical OR	If any of the two operands are non-zero (true) then condition becomes true.	(a or b) is True. (means 1)
not Logical NOT	Used to reverse the logical state of its operand.	<ul style="list-style-type: none">❖ Not (a and b) is True. (means True)❖ Not (a or b) is False. (means False)

PYTHON MEMBERSHIP OPERATORS

- Python's membership operators test for membership in a sequence, such as strings, lists, or tuples.
- There are two membership operators as explained below –

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1, if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1, if x is not a member of sequence y.

PYTHON IDENTITY OPERATORS

- Identity operators compare the memory locations of two objects.
- There are two Identity operators as explained below –

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1, if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1, if id(x) is not equal to id(y).

PYTHON OPERATORS PRECEDENCE

- The following table lists all operators from highest precedence to the lowest. Order - PEMDAS.

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisons, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

THANK YOU FOR ATTENDING THE SESSION!

Any Queries?