**\* PYTHON PROGRAMMING LANGUAGE BASIC INTRODUCTION & SYNTAX** \*

By *: MAHANKALI GOPINATH,B.E CSE* Sathyabama University,Chennai

Date:14/07/2019

* Python is a programming language.
* Python can be used on a server to create web applications.
* Example: print("Hello,World!")
* What is Python?
* Python is a popular programming language. It was created by Guido van Rossum,and released in 1991.

It is used for:

1)Web development (server-side),

2)Software development,

3)Mathematics

4)System scripting

* Key Points for Python Introduction and Uses :
* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written.
* This means that prototyping can be very quick.

Python can be treated in a procedural way, an object-orientated way or a functional way.

* Execution of Python Syntax:
* Python syntax can be executed by writing directly in the Command Line

SYNTAX:>>> print("Hello, World!")

Output: Hello, World!

* Or by creating a python file on the server, using the .py file extension(saving method)
* Python uses indentation to indicate a block of code.
* Example : if(5>2)

print("Five is greater than two!")

* **Python Variables**
* In Python variables are created the moment you assign a value to it
* Example:Variables in

X = 5

y = "Hello, World!"

* we need to discuss deeply about Variables,Let us look into it,
* **Creating Variables**
* Variables are containers for storing data values.
* Unlike other programming languages, Python has no command for declaring a variable.
* A variable is created the moment you first assign a value to it.
* Example: x = 5

y = "John"

print(x)

print(y)

* String variables can be declared either by using single or double quotes:

Example : x = "John"

# is the same as

x = 'John'

* **Assign Value to Multiple Variables**
* Python allows you to assign values to multiple variables in one line:
* Example : x, y, z = "Orange", "Banana", "Cherry"

print(x)

print(y)

print(z)

* And you can assign the same value to multiple variables in one line:

Example : x = y = z = "Orange"

print(x)

print(y)

print(z)

* **Output Variables**
* The Python print statement is often used to output variables.
* To combine both text and a variable, Python uses the + character:

Example : x = "awesome"

print("Python is " + x)

* You can also use the + character to add a variable to another variable:

Example : x = "Python is "

y = "awesome"

z = x + y

print(z)

* **Python Numbers**
* There are three numeric types in Python:
* int

float

complex

* Variables of numeric types are created when you assign a value to them

Example : x = 1 # int

y = 2.8 # float

z = 1j # complex

* To verify the type of any object in Python, use the type() function

Example : print(type(x))

print(type(y))

print(type(z))

* Int, or integer, is a whole number, positive or negative, without decimals, of

unlimited length.

* Float, or "floating point number" is a number, positive or negative, containing one or more decimals.
* >Float can also be scientific numbers with an "e" to indicate the power of 10.
* Complex numbers are written with a "j" as the imaginary part.
* **Type Conversion** : You can convert from one type to another with the int(), float(), and complex() methods:

Example : Convert from one type to another type

x = 1 # int

y = 2.8 # float

z = 1j # complex

#convert from int to float:

a = float(x)

#convert from float to int:

b =complex(y)

#convert from int to complex:

c = complex(x)

print(a)

print(b)

print(c)

print(type(a))

print(type(b))

print(type(c))

* You cannot convert complex numbers into another number type.
* **Assign String to a Variable**
* Assigning a string to a variable is done with the variable name followed by an equal sign and the string:
* Example :a = "Hello"

print(a)

* **Strings and Arrays**
* Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.
* However, Python does not have a character data type, a single character is simply a string with a length of 1.
* Square brackets can be used to access elements of the string.
* Example : Get the character at position 1 (remember that the first character has the position 0):

a = "Hello, World!"

print(a[1])

* Example : Substring. Get the characters from position 2 to position 5 (not included):

b = "Hello, World!"

print(b[2:5])

* Example : The strip() method removes any whitespace from the beginning or the end:

a = " Hello, World! "

print(a.strip()) # returns "Hello, World!"

* Example : The len() method returns the length of a string:

a = "Hello, World!"

print(len(a))

* Example : The lower() method returns the string in lower case:

a = "Hello, World!"

print(a.lower())

* Example : The upper() method returns the string in upper case:

a = "Hello, World!"

print(a.upper())

* Example : The replace() method replaces a string with another string:

a = "Hello, World!"

print(a.replace("H", "J"))

* Example : The split() method splits the string into substrings if it finds instances of the separator:

a = "Hello, World!"

print(a.split(","))

* **Combination of strings** : We can combine strings and numbers by using the format() method!
* The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

Example : Use the format() method to insert numbers into strings:

age = 36

txt = "My name is John, and I am {}"

print(txt.format(age))

* The format() method takes unlimited number of arguments, and are placed into the respective placeholders:
* Example : quantity = 3

itemno = 567

price = 49.95

myorder = "I want {} pieces of item {} for {} dollars."

print(myorder.format(quantity, itemno, price))

* You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

Example : quantity = 3

itemno = 567

price = 49.95

myorder = "I want to pay {2} dollars for {0} pieces of item {1}."

print(myorder.format(quantity, itemno, price))

* **Python Operators**
* Operators are used to perform operations on variables and values
* Python divides the operators in the following groups:

1) Arithmetic operators

2)Assignment operators

3)Comparison operators

4)Logical operators

5)Identity operators

6)Membership operators

7)Bitwise operators

8)Python Arithmetic Operators

* **Arithmetic operators**
* Arithmetic operators are used with numeric values to perform common mathematical operations:

**Operator Name Example**

+ Addition x + y

- Subtraction x - y

\* Multiplication x \* y

/ Division x / y

% Modulus x % y

\*\* Exponentiation x \*\* y

// Floor division x // y

* **Python Assignment Operators**
* Assignment operators are used to assign values to variables:

**Operator Example Same As**

= x = 5 x = 5

+= x += 3 x = x + 3

-= x -= 3 x = x - 3

\*= x \*= 3 x = x \* 3

**Operator** **Example Same As**

/= x /= 3 x = x / 3

%= x %= 3 x = x % 3

//= x //= 3 x = x // 3

\*\*= x \*\*= 3 x = x \*\* 3

&= x &= 3 x = x & 3

|= x |= 3 x = x | 3

^= x ^= 3 x = x ^ 3

>>= x >>= 3 x = x >> 3

<<= x <<= 3 x = x << 3

* **Python Comparison Operators**
* Comparison operators are used to compare two values:

**Operator**  **Name**

== Equal x == y

!= Not equal x != y

> Greater than x > y

< Less than x < y

>= Greater than or equal to x >= y

<= Less than or equal to x <= y

* **Python Logical Operators**
* Logical operators are used to combine conditional statements:

**Operator** **Description Example**

and Returns True if both statements are true x < 5 and x < 10

or Returns True if one of the statements is true x < 5 or x < 4

not Reverse the result, returns False if the result is tru (x < 5 and x < 10)

* **Python Identity Operators**
* Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

**Operator**  **Description** **Example**

is Returns true if both variables are the same object x is y

is not Returns true if both variables are not the same object x is not y

* **Python Membership Operators**
* Membership operators are used to test if a sequence is presented in an object:

**Operator** **Description**

in Returns True if a sequence with the specified value is present in the object

not in Returns True if a sequence with the specified value is not present in the object

* **Python Bitwise Operators**
* Bitwise operators are used to compare (binary) numbers:

**Operator** **Name**

& AND Sets each bit to 1 if both bits are 1

| OR Sets each bit to 1 if one of two bits is 1

^ XOR Sets each bit to 1 if only one of two bits is 1

~ NOT Inverts all the bits

<< Zero fill left shift Shift left by pushing zeros in from the right and let the leftmost bits fall off

>> Signed right shift Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

* **Collection of Arrays in Python :**
* Python Collections (Arrays)

There are four collection data types in the Python programming language:

1)List is a collection which is ordered and changeable. Allows duplicate members.

2)Tuple is a collection which is ordered and unchangeable. Allows duplicate members.

3)Set is a collection which is unordered and unindexed. No duplicate members.

4)Dictionary is a collection which is unordered, changeable and indexed. No duplicate members.

* Now we will discuss in Detail about the above mentioned ones,
* 1)LIST : A list is a collection which is ordered and changeable. In Python lists are written with square brackets.

Example : Create a List:

thislist = ["apple", "banana", "cherry"]

print(thislist)

* Access Items

You access the list items by referring to the index number:

Example : Print the second item of the list:

thislist = ["apple", "banana", "cherry"]

print(thislist[1])

* Change Item Value : To change the value of a specific item, refer to the index number:

Example : Change the second item:

thislist = ["apple", "banana", "cherry"]

thislist[1] = "blackcurrant"

print(thislist)

* Loop Through a List : You can loop through the list items by using a for loop:

Example : Print all items in the list, one by one:

thislist = ["apple", "banana", "cherry"]

for x in thislist:

print(x)

You will learn more about for loops in out Python For Loops Chapter under this section

* Check if Item Exists : To determine if a specified item is present in a list use the in keyword:

Example : Check if "apple" is present in the list:

thislist = ["apple", "banana", "cherry"]

if "apple" in thislist:

print("Yes, 'apple' is in the fruits list")

* List Length : To determine how many items a list has, use the len() method:

Example : Print the number of items in the list:

thislist = ["apple", "banana", "cherry"]

print(len(thislist))

Add Items

* To add an item to the end of the list, use the append() method:

Example : Using the append() method to append an item:

thislist = ["apple", "banana", "cherry"]

thislist.append("orange")

print(thislist)

* To add an item at the specified index, use the insert() method:

Example : Insert an item as the second position:

thislist = ["apple", "banana", "cherry"]

thislist.insert(1, "orange")

print(thislist)

* Remove Item :There are several methods to remove items from a list:

Example1 : The remove() method removes the specified item:

thislist = ["apple", "banana", "cherry"]

thislist.remove("banana")

print(thislist)

Example2 : The pop() method removes the specified index, (or the last item if index is not specified):

thislist = ["apple", "banana", "cherry"]

thislist.pop()

print(thislist)

Example3 : The del keyword removes the specified index:

thislist = ["apple", "banana", "cherry"]

del thislist[0]

print(thislist)

Example4 : The del keyword can also delete the list completely:

thislist = ["apple", "banana", "cherry"]

del thislist

Example5 : The clear() method empties the list:

thislist = ["apple", "banana", "cherry"]

thislist.clear()

print(thislist)

* **Copy a List**

You cannot copy a list simply by typing list2 = list1, because: list2 will only be a reference to list1, and changes made in list1 will automatically also be made in list2.

* There are ways to make a copy, one way is to use the built-in List method copy().

Example : Make a copy of a list with the copy() method:

thislist = ["apple", "banana", "cherry"]

mylist = thislist.copy()

print(mylist)

* Another way to make a copy is to use the built-in method list().

Example : Make a copy of a list with the list() method:

thislist = ["apple", "banana", "cherry"]

mylist = list(thislist)

print(mylist)

* **The list() Constructor**
* It is also possible to use the list() constructor to make a new list.

Example : Using the list() constructor to make a List:

thislist = list(("apple", "banana", "cherry")) # note the double round-brackets

print(thislist)

* **List Methods**

>Python has a set of built-in methods that you can use on lists.

**Method**  **Description**

append() Adds an element at the end of the list

clear() Removes all the elements from the list

copy() Returns a copy of the list

count() Returns the number of elements with the specified value

extend() Add the elements of a list (or any iterable), to the end of the current list

index() Returns the index of the first element with the specified value

insert() Adds an element at the specified position

pop() Removes the element at the specified position

remove() Removes the item with the specified value

reverse() Reverses the order of the list

sort() Sorts the list

* **PYTHON TUPLES**

>**Tuple** :A tuple is a collection which is ordered and unchangeable. In Python tuples are written with round brackets.

Example : Create a Tuple:

thistuple = ("apple", "banana", "cherry")

print(thistuple)

>Access Tuple Items : You can access tuple items by referring to the index number, inside square brackets:

Example : Return the item in position 1:

thistuple = ("apple", "banana", "cherry")

print(thistuple[1])

>Change Tuple Values : Once a tuple is created, you cannot change its values. Tuples are unchangeable.

>Loop Through a Tuple : You can loop through the tuple items by using a for loop.

Example : Iterate through the items and print the values:

thistuple = ("apple", "banana", "cherry")

for x in thistuple:

print(x)

You will learn more about for loops in out Python For Loops Chapter.

> To Check if Item Exists : To determine if a specified item is present in a tuple use the in keyword:

Example : Check if "apple" is present in the tuple:

thistuple = ("apple", "banana", "cherry")

if "apple" in thistuple:

print("Yes, 'apple' is in the fruits tuple")

* **Tuple Length** : To determine how many items a tuple has, use the len() method:

Example : Print the number of items in the tuple:

thistuple = ("apple", "banana", "cherry")

print(len(thistuple))

* **Add Items** : Once a tuple is created, you cannot add items to it. Tuples are unchangeable.

* **Remove Items** :

Note: You cannot remove items in a tuple.

Tuples are unchangeable, so you cannot remove items from it, but you can delete the tuple completely:

Example : The del keyword can delete the tuple completely:

thistuple = ("apple", "banana", "cherry")

del thistuple

print(thistuple) #this will raise an error because the tuple no longer exists

* **Tuple Methods** : Python has two built-in methods that you can use on tuples.

**Method**  **Description**

count() Returns the number of times a specified value occurs in a tuple

index() Searches the tuple for a specified value and returns the position of where it was found

* **PYTHON SETS**

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.

Example : Create a Set:

thisset = {"apple", "banana", "cherry"}

print(thisset)

#Note: Sets are unordered, so the items will appear in a random order.

* **Access Items** : You cannot access items in a set by referring to an index, since sets are unordered the items has no index.
* But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

Example 1 : Loop through the set, and print the values:

thisset = {"apple", "banana", "cherry"}

for x in thisset:

print(x)

Example 2 : Check if "banana" is present in the set:

thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)

* **Change Items** : Once a set is created, you cannot change its items, but you can add new items.
* **Add Items** : To add one item to a set use the add() method.

To add more than one item to a set use the update() method.

Example1 : Add an item to a set, using the add() method:

thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)

Example2 : Add multiple items to a set, using the update() method:

thisset = {"apple", "banana", "cherry"}

thisset.update(["orange", "mango", "grapes"])

print(thisset)

* **Get the Length of a Set** : To determine how many items a set has, use the len() method.

Example : Get the number of items in a set:

thisset = {"apple", "banana", "cherry"}

print(len(thisset))

* **Remove Item** : To remove an item in a set, use the remove(), or the discard() method.

Example1 : Remove "banana" by using the remove() method:

thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)

#Note: If the item to remove does not exist, remove() will raise an error.

Example2 : Remove "banana" by using the discard() method:

thisset = {"apple", "banana", "cherry"}

thisset.discard("banana")

print(thisset)

#Note: If the item to remove does not exist, discard() will NOT raise an error.

* You can also use the pop(), method to remove an item, but this method will remove the last item. Remember that sets are unordered, so you will not know what item that gets removed.
* The return value of the pop() method is the removed item.

Example1 : Remove the last item by using the pop() method:

thisset = {"apple", "banana", "cherry"}

x = thisset.pop()

print(x)

print(thisset)

#Note: Sets are unordered, so when using the pop() method, you will not know which item that gets removed.

Example2 : The clear() method empties the set:

thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset)

Example3 : The del keyword will delete the set completely:

thisset = {"apple", "banana", "cherry"}

del thisset

print(thisset)

* **SYNOPSIS FOR SET METHODS**
* Python has a set of built-in methods that you can use on sets.

**Method** **Description**

add() Adds an element to the set

clear() Removes all the elements from the set

copy() Returns a copy of the set

difference() Returns a set containing the difference between two or more sets

difference\_update() Removes the items in this set that are also included in another, specified set

discard() Remove the specified item

intersection() Returns a set, that is the intersection of two other sets

intersection\_update() Removes the items in this set that are not present in other,

isdisjoint() Returns whether two sets have a intersection or not

issubset() Returns whether another set contains this set or not

issuperset() Returns whether this set contains another set or not

pop() Removes an element from the set

remov() Removes the specified element

symmetric\_difference() Returns a set with the symmetric differences of two sets

symmetric\_difference\_update() Inserts the symmetric differences from this set and another

union() Return a set containing the union of sets

update() Update the set with the union of this set and others

* **PYTHON DICTIONARIES**
* **Dictionary** : A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

Example : Create and print a dictionary:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

print(thisdict)

* **Accessing Items** : You can access the items of a dictionary by referring to its key name, inside square brackets:

Example : Get the value of the "model" key:

x = thisdict["model"]

There is also a method called get() that will give you the same result:

Example : Get the value of the "model" key:

x = thisdict.get("model")

* **Change Values** : You can change the value of a specific item by referring to its key name:

Example : Change the "year" to 2018:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict["year"] = 2018

* **Loop Through a Dictionary** :

You can loop through a dictionary by using a for loop.

When looping through a dictionary, the return value are the keys of the dictionary, but there are methods to return the values as well.

Example1 : Print all key names in the dictionary, one by one:

for x in thisdict:

print(x)

Example2 : Print all values in the dictionary, one by one:

for x in thisdict:

print(thisdict[x])

Example3 : You can also use the values() function to return values of a dictionary:

for x in thisdict.values():

print(x)

Example4 : Loop through both keys and values, by using the items() function:

for x, y in thisdict.items():

print(x, y)

* **Check if Key Exists** : To determine if a specified key is present in a dictionary use the in keyword:

Example : Check if "model" is present in the dictionary:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

if "model" in thisdict:

print("Yes, 'model' is one of the keys in the thisdict dictionary")

* **Dictionary Length** : To determine how many items (key-value pairs) a dictionary has, use the len() method.

Example : Print the number of items in the dictionary:

print(len(thisdict))

* **Adding Items** : Adding an item to the dictionary is done by using a new index key and assigning a value to it:

Example : thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict["color"] = "red"

print(thisdict)

* **Removing Items** : There are several methods to remove items from a dictionary:

Example1 : The pop() method removes the item with the specified key name:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.pop("model")

print(thisdict)

Example2 : The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead):

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.popitem()

print(thisdict)

Example3 : The del keyword removes the item with the specified key name:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

del thisdict["model"]

print(thisdict)

Example4 : The del keyword can also delete the dictionary completely:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

del thisdict

print(thisdict) #this will cause an error because "thisdict" no longer exists.

Example5 : The clear() keyword empties the dictionary:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.clear()

print(thisdict)

* **Copy a Dictionary**
* You cannot copy a dictionary simply by typing dict2 = dict1, because: dict2 will only be a reference to dict1, and changes made in dict1 will automatically also be made in dict2.
* There are ways to make a copy, one way is to use the built-in Dictionary method copy().

Example : Make a copy of a dictionary with the copy() method:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

mydict = thisdict.copy()

print(mydict)

* Another way to make a copy is to use the built-in method dict().

Example : Make a copy of a dictionary with the dict() method:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

mydict = dict(thisdict)

print(mydict)

* **The dict() Constructor** : It is also possible to use the dict() constructor to make a new dictionary:

Example : thisdict = dict(brand="Ford", model="Mustang", year=1964)

# note that keywords are not string literals

# note the use of equals rather than colon for the assignment

print(thisdict)

* **Dictionary Methods** :
* Python has a set of built-in methods that you can use on dictionaries.

**Method Description**

clear() Removes all the elements from the dictionary

copy() Returns a copy of the dictionary

fromkeys() Returns a dictionary with the specified keys and values

get() Returns the value of the specified key

items( ) Returns a list containing the a tuple for each key value pair

keys() Returns a list containing the dictionary's keys

pop() Removes the element with the specified key

popitem() Removes the last inserted key-value pair

setdefault() Returns the value of the specified key. If the key does not exist: insert the key, with the specified value

update() Updates the dictionary with the specified key-value pairs

values() Returns a list of all the values in the dictionary

**\*\*References / Links for More Learning :**

1)https://www.coursera.org/courses?query=python

2)https://www.codecademy.com/learn/learn-python

3)https://www.edx.org/learn/python

4)https://www.learnpython.org/

5)https://www.udemy.com/topic/python/free/

6)https://www.udacity.com/course/introduction-to-python--ud1110

7)https://alison.com/course/introduction-to-programming-with-python

*For more course links,mail to us for details*

---------------------------------------------Notes will be Continued with Next Part -----------------------------------------

Gathered and Edited by **Mahankali Gopinath**,B.E CSE Sathyabama University

Contact Details : Email Address : mahankaligopinath382@gmail.com

Website/Blog : https://gopinathmahankali.blogspot.com/

Facebook : https://www.facebook.com/gopi.cherry382

Twitter : https://twitter.com/gopinath\_382?lang=en

THE END