Author:

Muhammad Naveed

Contact-Email:

naveedali786aziz@gmail.com

> **Basics**
>
> Basic syntax from the python programming language
>
> **Showing Output To User**

> The print function is used to display or print output

```
print("Content that you wanna print on screen")
```

> we can display the content present in object using prit function as follows:-

```
name = "Naveed"
print("Hi my name is: ",name)
```

> **Taking Input From User**

> The input function is used to take input from the user

```
name=input("Enter your name: ")
print('My name is:',name)
```

> To take input in form of other datatypes we need to typecaste them as follows

```
print(var1)
```

> Similar we take input as an integer:

```
var1=float(input("enter the float value"))
print(var1)
```

Range Function

Range function returns a sequence of numbers, eg, numbers starting from 0 to n-1 for range(0, n) For Example: Display all even numbers between 1 to 100

```
for i in range(0,101,2):
       print(i)
```

Comments

Comments are used to make the code more understandable for programmers, and they are not executed by compiler or interpreter.

- Single line comment (#)
- Multi-line commen (ctrl +/)

Single line comment

```
#This is a single line comment
```

Multi-line comment

```
'''This is a
multi-line
comment'''
```

Escape Sequence

An escape sequence is a sequence of characters,it doesn't represent itself(but is translated into another character) when used inside string literal or character.

Newline

Newline Character

```
\n
```

Backslash

It adds a backslash

```
print("\\")
```

Single Quote

It adds a single quotation mark

```
print("\'")
```

Tab

It gives a tab space

```
\t
```

Backspace

It adds a backspace

```
\b
```

Octal value

It represents the value of an octal number

```
\ooo
```

Hex value

It represents the value of a hex number

```
\xhh
```

Strings
Python string is a sequence of characters, and each character can be individually accessed. Using its index.

String

You can create Strings by enclosing text in both forms of quotes - single quotes or double-quotes.

```
variable_name ="it is variable"
```

Example

```
var1="Naveed"
print("string is ",var1)
```

Indexing

The position of every character placed in the string starts from 0th position ans step by step it ends at length-1 position

Slicing

Slicing refers to obtaining a sub-string from the given string.

```
var_name\[1 : 5]
```

isalnum() method

Returns True if all characters in the string are alphanumeric

```
string_variable.isalnum()
```

isalpha() method

Returns True if all characters in the string are alphabet

```
string_variable.isalpha()
```

isdecimal() method

Returns True if all characters in the string are decimals

```
string_variable.isdecimal()
```

isdigit() method

Returns True if all characters in the string are digits

```
string_variable.isdigit()
```

islower() method

Returns True if all characters in the string are lower case

```
string_variable.islower()
```

isspace() method

Returns True if all characters in the string are whitespaces

```
string_variable.isspace()
```

isupper() method

Returns True if all characters in the string are upper case

```
string_variable.isupper()
```

lower() method

Converts a string into lower case

```
string_variable.lower()
```

upper() method

Converts a string into upper case

```
string_variable.upper()
```

strip() method

It removes leading and trailing spaces in the string

```
string_variable.strip()
```

List

A List in Python represents a list of comma-separated values of any data type between square brackets.

List

```
var_name =[element1, element2,...]
```

Indexing of List

The position of every elements placed in the string starts from 0th position ans step by step it ends at length-1 position

Empty List

```
my_list=[]
```

Index Method

Returns the index of the first element with the specified value List is ordered,indexed,mutable and most flexible and dynamic collection of elements in

python.

```
my_list.index(element)
```

Append Method

Adds an element at the end of the list

```
list.append(element)
```

Extend Method

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

```
list.extend(iterable)
```

Insert Method

Adds an element at the specified position

```
list.insert(position, element)
```

POP Method

Removes the element at the specified position and returns it

```
list.pop(position)
```

Remove Method

The remove( ) method removes the first occurrence of a given item from the list

```
list.remove(element)
```

Clear Method

Removes all the elements from the list

```
list.clear()
```

Count Method

Returns the number of elements with the specified value

```
list.count(value)
```

Reverse Method

Reverse the order of the list

```
list.reverse()
```

Sort Method

Sorts the list

```
list.sort(reverse=True\|False)
```

Clear Method Removes all the elements from the list

```
list.clear()
```

Count Method Returns the number of elements with the specified value

```
list.count(value)
```

Reverse Method Reverses the order of the list

```
list.reverse()
```

Remove Method

The remove() method removes the first occurrence of a given item from the list

```
list.remove(element)
```

Tuples

Tuples are represented as a list of comma-separated values of any data type within parentheses.

Tuple Creation

```
variable_name =(element1, element2,...)
```

These elements can be of different datatypes

Indexing of Tuples

The position of every elements placed in the string starts from 0th position ans step by step it ends at length-1 position Tuples are ordered,indexing,immutable and most secured collection of elements

Lets talk about some of the tuple methods:

Tuple Methods

Count Method

It returns the number of times a specified value occurs in a tuple

```
tuple.count(value)
```

Index Method

It searches the tuple for a specified value and returns the position.

```
tuple.index(value)
```

Sets

A set is a collection of multiple values which is both unordered and unindexed. It is written in curly brackets.

Set Creation: Way 1

```
var_name ={element1, element2,...}
```

Set Creation: Way 2

```
var_name =set(\[element1, element2,...\])
```

Set Methods:

Lets talk about some of the methods of sets:

Add() Method

Adds an element to a set

```
set.add(element)
```

Clear() Method

Remove all elements from a set

```
set.clear()
```

Discard() Method

Removes the specified item from the set

```
set.discard(value)
```

Intersection() Method

Returns intersection of two or more sets

```
set.intersection(set1, set2 ... etc)
```

issubset() Method

Checks if a Set is Subset of Another Set

```
set.issubset(set)
```

POP() Method

Removes an element from the set

```
set.pop()
```

Remove() Method

Removes the specified element from the Set

```
set.remove(item)
```

Union() Method

Returns the union of Sets

```
set.union(set1, set2...)
```

Dictionaries

The dictionary is an unordered set of comma-separated key: value pairs, within {}, with the

requirement that within a dictionary, no two keys can be the same.

Dictionary The dictionary is an unordered set of comma-separated key:value pairs, within {}, with the requirement that within a dictionary, no two keys can be the same

Dictionary

```
<dictionary-name> = {<key>: value, <key>: value ...}
```

Dictionary is ordered and mutable collection of elements.Dictionary allows duplicate values but not duplicate keys.

Empty Dictionary

By putting two curly braces, you can create a blank dictionary

```
mydict={}
```

Adding Element to a Dictionary

By this method, one can add new elements to the dictionary

```
<dictionary\>\[\<key\>\]=\<value\>
```

Updating Element in a Dictionary

If the specified key already exists, then its value will get updated

```
<dictionary\>\[\<key\>\]=\<value\>
```

Deleting Element from a dictionary

Del let to delete specified key: value pair from the dictionary

```
del <dictionary\>\[\<key\>\]
```

Dictionary Functions & Methods len() method

It returns the length of the dictionary, i.e., the count of elements (key: value pairs) in the

len() method

It returns the length of the dictionary, i.e., the count of elements (key: value pairs) in the dictionary

```
len(dictionary)
```

Clear() Method

Removes all the elements from the dictionary

```
dictionary.clear()
```

Get() Method

Returns the value of the specified key

```
dictionary.get(keyname)
```

Items() Method

Returns a list containing a tuple for each key-value pair

```
dictionary.items()
```

keys() Method

Returns a list containing the dictionary's keys

```
dictionary.keys()
```

Values() Method

Returns a list of all the values in the dictionary

```
dictionary.values()
```

Update() Method

Updates the dictionary with the specified key-value pairs

```
dictionary.update(iterable)
```

Indentation

In Python, indentation means the code is written with some spaces or tabs into many different blocks of code to indent it so that the interpreter can easily execute the Python code.

Indentation is applied on conditional statements and loop control statements. Indent specifies the block of code that is to be executed depending on the conditions.

Conditional Statements

The if statements are the conditional statements in Python, and these implement selection constructs (decision constructs).

if Statement

```
if(conditional expression):
    statements
```

if-else Statement

```
if(conditional expression):
  statements
else:
  statements
```

if-elif Statement

```
if(conditional expression):
  statements
elif(conditional expression):
  statements
else:
  statements
```

Nested if-else Statement

```
if(conditional expression):
  if(conditional expression):
```

```
    statements
else:
  statements
else:
  statements
```

Iterative Statements

An iteration statement, or loop, repeatedly executes a statement, known as the loop body, until the controlling expression is false (0).

Example

```
a=15
b=20
c=12
if(a>b and a>c):
      print(a,"is greatest")
elif(b>c and b>a):
      print(b," is greatest")
else:
    print(c,"is greatest")
```

For Loop

The for loop of Python is designed to process the items of any sequence, such as a list or a string, one by one.

```
for\<variable\>in\<sequence\>:
statements_to_repeat
```

For Example

```
for i in range(1,101,1):
           print(i)
```

While Loop

A while loop is a conditional loop that will repeat the instructions within itself as long as a conditional remains true.

```
while\<logical-expression\>:
```

```
loop-body
```

For Example

```
i=1
while(i<=100):
        print(i)
        i=i+1
```

Break Statement

The break statement enables a program to skip over a part of the code. A break statement terminates the very loop it lies within.

```
for\<var\>in\<sequence\>:
statement1
if\<condition\>:
break
statement2
statement_after_loop
```

For Example

```
for i in range(1,101,1):
    print(i ,end=" ")
    if(i==50):
        break
    else:
        print("Mississippi")
print("Thank you")
```

Continue Statement

The continue statement skips the rest of the loop statements and causes the next iteration to occur.

```
for\<var\>in\<sequence\>:
statement1
if\<condition\>:
continue
statement2
```

```
statement3
statement4
```

For Example

```
for i in [2,3,4,6,8,0]:
    if (i%2!=0):
        continue
    print(i)
```

Functions

A function is a block of code that performs a specific task. You can pass parameters into a function. It helps us to make our code more organized and manageable.

Function Creation

```
defmy_function(parameters):
\# Statements
```

def keyword is used before defining the function.

Function Call

```
my_function()
```

Whenever we need that block of code in our program simply call that function name whenever neeeded. If parameters are passed during defing the function we have to pass the parameters while calling that function

For Example

```
def add():  #function defination
        a=10
        b=20
       print(a+b)
add()   #function call
```

Return statement in Python function

The function return statement return the specified value or data item to the caller.

```
    return [value/expression]
```

Arguments in python function

Arguments are the values passed inside the parenthesis of the function while defining as well as while calling.

```
def my_function(arg1,arg2,arg3....argn):
          #statements
my_function(arg1,arg2,arg3....argn)
```

Example

```
def add(a,b):
      return a+b
x=add(7,8)
print(x)
```

File Handling

File handling refers to reading or writing data from files. Python provides some functions that allow us to manipulate data in the files.

Open() Function

```
var_name =open("file name","opening mode")
```

Close() Function

```
var_name.close()
```

Read () Function

The read functions contains different methods, read(),readline() and readlines()

```
read()#return one big string
```

It returns a list of lines

```
readlines() #returns a list
```

It returns one line at a time

```
readline
```

Write () Function

This function writes a sequence of strings to the file.

```
write ()#Used to write a fixed sequence of characters to a file
```

It is used to write a list of strings

```
writelines()
```

Exception Handling
An exception is an unusual condition that results in an interruption in the flow of the program.

Try and Except

A basic try-catch block in python. When the try block throws an error, the control goes to the except block.

```
try:
    [Statement body block]
    raise Exception()
except Exceptionname:
    [Error processing block]
```

else

The else block is executed if the try block have not raise any exception and code had been running successfully

```
try:
     #statements
except:
     #statements
else:
      #statements
```

Finally

> Finally block will be executed even if try block of code has been running successsfully or except block of code is been executed. finally block of code will be executed compulsory

Start coding or generate with AI.