Python 3 Reference Document

1. Variable

Syntax:

```
variable_name = value
```

Example:

```
foo = 2
foo_bar="hello world"
```

2. PRINT STATEMENT

Syntax:

```
print(value/variable)
```

Example:

```
print("Foo Bar")
```

3. SELECTION

3.1. IF

Syntax:

```
if(condition):
   #block of statements
```

Example:

```
if(foo<3):
    print("foo is less than 3")
```

3.2. IF ELSE

Syntax:

```
if(condition):
    #block of statements
else:
    #block of statements
```

Example:

```
if(foo > 3):
    print("foo is greater than 3")
else:
    print("foo is less than 3")
```

3.3. ELIF LADDER

Syntax:

```
if(condition):
    #block of statements
elif(condition):
    #block of statements
else:
    #block of statements
```

Example:

```
if(foo == 1):
    print("foo equals 1")
elif(foo == 2):
    print("foo equals 2")
else:
    print("foo value is other than 1 and 2")
```

3.4. NESTED IF

Syntax:

```
if(condition):
    #block of statements
    if(condition):
        #block of statements
    else:
        #block of statements
else:
    #block of statements
```

Example:

```
if(foo > 0):
    if(foo > 30):
        print("foo is greater than 30")
    else:
        print("foo is not greater than 30")
else:
    print("foo is not greater than 0")
```

4. ITERATION

4.1. WHILE LOOP

Syntax:

```
while(condition):
    #block of statements
```

Example:

```
foo = 2
while(foo<=5):
    print(foo)
```

```
    foo = foo+1
```

4.2. FOR LOOP

Syntax-1:

```
for <variable> in <sequence>:
    #block of statements
```

Example-1:

```
for number in 1,2,3,4,5:
    print(number)
```

Syntax-2:

```
for number in range(x,y):
    #block of statements
```

Example-2:

```
foo_bar=('Apple','Banana','Mango')
for index in range(0,len(foo_bar)):
    print(foo_bar[index])
```

5. BREAK

Syntax:

```
break
```

Example:

```
for letter in "PYTHON":
    if(letter == "H"):
        break
    print(letter)
```

6. CONTINUE

Syntax:

```
continue
```

Example:

```
for letter in "PYTHON":
    if(letter == "H"):
        continue
    print(letter)
```

7. LIST

Syntax:

```
sample_list= []
```

Example:

```
foo_bar= [1,2,3,4]
```

7.1. APPEND

Syntax:

```
sample_list.append(element)
```

Example:

```
foo_bar= [1,2,3,4]
foo_bar.append(5)
```

7.2. INSERT

Syntax:

```
sample_list.insert(index_position,element)
```

Example:

```
foo_bar= [1,2,3,4]
foo_bar.insert(3,6)
```

7.3. POP

Syntax:

```
sample_list.pop(index)
```

Example:

```
foo_bar= [1,2,3,4]
foo_bar.pop(3)
```

7.4. REMOVE

Syntax:

```
sample_list.remove(element)
```

Example:

```
foo_bar= [1,2,3,4]
foo_bar.remove(4)
```

7.5. SORT

Syntax:

```
sample_list.sort()
```

Example:

```
foo_bar= [1,2,3,4]
foo_bar.sort()
```

7.6. REVERSE

Syntax:

```
sample_list.reverse()
```

Example:

```
foo_bar= [1,2,3,4]
foo_bar.reverse()
```

7.7. SLICE

Syntax:

```
sample_list.slice[start_position:end_position]
```

Example:

```
foo_bar= [1,2,3,4]
foo_bar[1:3]
```

8. TUPLE

Syntax:

```
tuple_name=(value1,value2,…value n)
```

Example:

```
foo=("Moto","Apple","Sony")
```

9. DICTIONARY

Syntax:

```
#Dictionary declaration
dict_name={key1:value1, key2:value2,….        key n:value n}

#Dictionary value updating
dict_name.update(dict_name1)

#Getting the value for a given key
dict_name.get(key1)
```

Example:

```
foo={"Name":"Maddy","Age":18}
print(foo.get("Name"))
foo_bar={"Address":"India"}
foo.update(foo_bar)
```

10. LIBRARIES

10.1. STRING

Syntax:

```
variable.count("count_of_string_to_find")
variable.replace("old_string","new_string")
variable.find("string_to_find")
variable.startswith("string_to_match")
variable. endswith("string_to_match")
```

```
variable.isdigit()
variable.upper()
variable.lower()
variable.split("string_based_on_split")
variable[start_position:end_position]
```

Example:

```
foo="I love python"
foo.count("o")
foo.replace("l","L")
foo.find("python")
foo.startswith("I")
foo. endswith("on")
foo.isdigit()
foo.upper()
foo.lower()
foo.split(" ")
foo[1:4]
```

10.2. RANDOM

Syntax:

```
import random
random.randrange(lower_limit,upper_limit)
```

Example:

```
import random
random.randrange(10,50)
```

10.3. TIME

Syntax:

```
import time
time.gmtime()
time.localtime()
time.timezone
```

Example:

```
import time
print(time.gmtime())
print(time.localtime())
print(time.timezone)
```

10.4. MATH

Syntax:

```
import math
math.ceil(decimal_value)
math.floor(decimal_value)
math.factorial(value)
math.fabs(decimal_value)
```

Example:

```
import math
print(math.ceil(9.6))
print(math.floor(9.6))
print(math.factorial(5))
print(math.fabs(9.6))
```

11. EXCEPTION

11.1 TRY-EXCEPT

Syntax:

```
try:
    #block of statements
except:
    #If there is any exception, then execute this block
```

Example:

```
try:
    foo = 100/0
except:
    print("Number cannot be divisible by 0")
```

11.2. TRY-EXCEPT-FINALLY

Syntax:

```
try:
    #block of statements
except:
    #If there is any exception, then execute this block
finally:
    #This would always be executed
```

Example:

```
try:
    foo = 100/0
except:
    print("Number cannot be divisible by 0")

finally:
    print("Program is terminating")
```

12. FUNCTION

Syntax:

```
def function_name(parameters):# Function declaration
            #function body
            [return]

function_name(values) # Function call
```

Example:

```
def  sum(foo,foo_bar):
```

```
    print(foo+foo_bar)


sum(5,5)
```

12.1. POSITIONAL ARGUMENTS

Syntax:

```
def function_name(parameter1,parameter2):
                #function body
                [return]


function_name(value1,value2)
```

Example:

```
def  sum(foo,foo_bar):
                print(foo+foo_bar)

sum(10,10)
```

12.2. KEYWORD ARGUMENTS

Syntax:

```
def function_name(parameter1,parameter2):
                #function body
                [return]


function_name(parameter1=value1,parameter2=value2)
```

Example:

```
def  sum(foo,foo_bar):
                print(foo+foo_bar)

sum(foo_bar=10,foo=5)
#(or)
sum(foo=5,foo_bar=10)
```

12.3. DEFAULT ARGUMENTS

Syntax:

```
def function_name(parameter1,parameter2=value):
    #Function body
    [return]

function_name(value1)
```

Example:

```
def  sum(foo,foo_bar=10):
    print(foo+foo_bar)
```

```
sum(2)
#(or)
sum(2,4)
```

12.4. VARIABLE NUMBER OF ARGUMENTS

Syntax:

```
def function_name(*variable_tuple):
    #Function body
    [return]

function_name(value1/value1,value2,…valuen)
```

Example:

```
def  sum(*foo):
    foo_bar=0
    for i in foo:
        foo_bar+=i
    print(foo_bar)

sum(2,4,6)
#(or)
sum(1,2)
```

13. VARIABLE SCOPE

13.1. GLOBAL VARIABLE

Syntax:

```
variable1=value            #Global variable, can be accessible anywhere.

def function_name():
    #function body
    [return]
```

Example:

```
foo=100

def function1():
    global foo
    foo+=1


print(foo)
function1()
print(foo)
```

13.2. LOCAL VARIABLE

Syntax:

```
def function_name():
```

```
    variable1=value #Local variable, can accessible only inside this function.
```

Example:

```
def function1():
    foo=100
    foo+=1
    print(foo)

function1()
print(foo) #This statement will give an error as variable,foo is  local to
```

14. PACKAGE

Syntax:

```
from packagename import modulename
#(or)
import packagename.modulename
```

Example:

```
from Flights import ManageFlights
#(or)
import Flights.ManageFlights
```

15. FILE HANDLING

15.1. OPENING A FILE

Syntax:

```
file = open(file_name [,access_mode])
```

Example:

```
sample_file=open(sample.txt,r)
```

15.2. CLOSING A FILE

Syntax:

```
close(file_name)
```

Example:

```
close(sample.txt)
```

15.3. WRITING INTO A FILE

Syntax:

```
file.write(string)
```

Example:

```
sample_file.write("Welcome to files…")
```

15.4. READING FROM A FILE

Syntax:

```
file.read()
```

Example:

```
sample_file.read()
```

16. REGULAR EXPRESSIONS

Example:

```
re.search(r"come","Welcome")
Output: come

re.search(r"c..e","Welcome")
Output: come

re.search(r"c\dme","Welc0me")
Output: c0me

re.search(r"W[0-9]e","W2elcome")
Output: W2e

re.search(r"Wel|Fel","Welcome")
Output: Wel

re.search(r"Welcome\s","Welcome to Regular Expression")
Output: Welcome #Will check whether space is there after "Welcome"

re.search(r"e$","Welcome")
Output: e

re.search(r"^W","Welcome")
Output: W

re.sub(r"Felcome",r"Welcome","Felcome to Regular Expression")
Output: Welcome to Regular Expression
```

17. LAMBDA EXPRESSIONS

Syntax:

```
lambda_name = lambda variable 1, variable 2,…variable n : lambda_operation
```

Example:

```
sum = lambda foo, foo_bar : foo + foo_bar
print(sum(3,3))
```

18. ITERATORS

Example:

```
'''
printing list data
'''
```

```
list=[10,2,100,5]

for i in range(0,len(list)):
    print(list[i])

print("--------------------")
'''
printing list data
'''
list=[10,2,100,5]
for i in range(0,len(list)):
    print(list[i])

print("--------------------")
'''
printing characters of string
'''
name="INFOSYS"
for char in name:
    print(char)
'''
printing characters of string
'''
name="INFOSYS"
for char in "INFOSYS":
    print(char)

dict={"a":100,"b":500,"c":300}

'''
get all keys from the dictionory
'''
list=dict.keys()
print(list)
dict={"a":100,"b":500,"c":300}

'''
iterating through the dictionary
'''

for key in dict:
    print(key)
    print(dict[key])
dict={"a":100,"b":500,"c":300}

'''
iterating through the dictionary using .items()
'''

for key,value in dict.items():
    print(key,value)
```