While loop : 'INCREMENTAL' to Overcome 'INFINITY LOOP'

PAGE 97

In [1]:
```
x = 1
while x<=10:
    print(x)
    x+=1
```

```
1
2
3
4
5
6
7
8
9
10
```

In [2]:
```
x = 1
while x<=10:
    print(x)
    x= x+1
```

```
1
2
3
4
5
6
7
8
9
10
```

In [5]:
```
def print_name(name):
    """This function prints name"""
    print("Hello...! " + str(name))
```

In [17]:
```
print_name("\tprasanth")
```

```
Hello...!       prasanth
```

In [34]:
```
password = 'london12345!'
if len(password) >=11 and '!' in password:
    print('password correct')
else:
    print('password incorrect')
```

```
password correct
```

In [37]:
```
password = 'london12345'
if len(password)>=11 and '!' in password:
    print('password correct')
else:
    print('password incorrect')
```

```
password incorrect
```

In [38]:
```
x = list(range(2,11,2))
x
```

Out[38]:
```
[2, 4, 6, 8, 10]
```

In [40]:
```
index_count = 0
for letter in 'abcde':
    print('at index{} the letter is {}'.format(index_count,letter))
```

```
at index0 the letter is a
at index0 the letter is b
at index0 the letter is c
at index0 the letter is d
at index0 the letter is e
```

In [41]:
```
index_count = 0
for letter in 'abcde':
    print('at index{} the letter is {}'.format(index_count,letter))
    index_count += 1
```

```
at index0 the letter is a
at index1 the letter is b
at index2 the letter is c
at index3 the letter is d
at index4 the letter is e
```

1. Incremental:

page 97

In [3]:
```
cnt = 2
while cnt<6:
    cnt+=1
    print("This is inside the loop")
else:
    print("This is outside the loop")
    print(cnt)
```

```
This is inside the loop
This is inside the loop
This is inside the loop
This is inside the loop
This is outside the loop
6
```

2. while else: 'else statement in the while loop'

page 97

In [18]:
```
cnt = 2
while cnt <6:
    print(cnt)
    cnt+=1
    print("This is inside the loop")
else:
    print("This is outside the loop")
    print(cnt)
```

```
2
This is inside the loop
3
This is inside the loop
4
This is inside the loop
5
This is inside the loop
This is outside the loop
6
```

In [20]:
```
# page 98 Break and continue in conditional loop:
# Break - stop the loop:
# continue - it skip current recurrence and will continue
```

In [21]:
```
for number in range (1,10):
    if number == 7:
        break
    print(number)
```

```
1
2
3
4
5
6
```

In [22]:
```
for number in range (1,10):
    if number == 7:
        continue
    print(number)
```

```
1
2
3
4
5
6
8
9
```

In [24]:
```
# page 99 Nested loop : Break it's an a pure nested loop
```

In [26]:
```
list1 = [4,5,6,7]
list2 = [10,20,30,40]
for i in list1:
    for j in list2:
        if j ==20:
            break
        print(i*j)
    print("Outside the Nested loop")
```

```
40
Outside the Nested loop
50
Outside the Nested loop
60
Outside the Nested loop
70
Outside the Nested loop
```

In [27]:
```
list1 = [4,5,6,7]
list2 = [10,20,30,40]
for i in list1:
    for j in list2:
        if j ==20:
            continue
        print(i*j)
    print("Outside the Nested loop")
```

```
40
120
160
Outside the Nested loop
50
150
200
Outside the Nested loop
60
180
240
Outside the Nested loop
70
210
280
Outside the Nested loop
```

In [1]:
```
x = list (range(2,11,2))
x
```

Out[1]:
```
[2, 4, 6, 8, 10]
```

In [3]:
```
index_count = 0
for letter in 'abcde':
    print('At index {} the letter is {}'.format(index_count,letter))
```

```
At index 0 the letter is a
At index 0 the letter is b
At index 0 the letter is c
At index 0 the letter is d
At index 0 the letter is e
```

In [5]:
```
# page 101 count the iterations manually:
# here we are counting the index manually

index_count = 0
for letter in 'abcde':
    print('At index {} the letter is {}'.format(index_count,letter))
    index_count += 1
```

```
At index 0 the letter is a
At index 1 the letter is b
At index 2 the letter is c
At index 3 the letter is d
At index 4 the letter is e
```

In [8]:
```
# page 102 Enumerate():
# Enumerate count the iteration automatically:
# Enumerate helps us to count the number of iterations:
```

In [9]:
```
word = 'abcde'
for item in enumerate(word):
    print(item)
```

```
(0, 'a')
(1, 'b')
(2, 'c')
(3, 'd')
(4, 'e')
```

In [10]:
```
# split independent: enumerate():
#page 103
```

In [18]:
```
word = 'abcde'
for index,letter in enumerate (word):
    print(index)
    print(letter)
    print('\n\n')
```

```
0
a



1
b



2
c



3
d



4
e
```

In [19]:
```
# zip():
```

In [20]:
```
list1 = [1,2,3,4]
list2 = ['a','b','c','d']
for item in zip(list1,list2):
    print(item)
```

```
(1, 'a')
(2, 'b')
(3, 'c')
(4, 'd')
```

In [21]:
```
# page 104 while loop :
```

In [30]:
```
python = "i am learning Python, "
python += "it is easy language. "
python += "it is very flexible"
```

In [31]:
```
python
```

Out[31]:
```
'i am learning Python, it is easy language. it is very flexible'
```

In [32]:
```
# page 105
# logic : whatever you type will repeat same:
```

In [36]:
```
prompt = "\n Hi i'm Prasanth, Please tell me something"
prompt += "\n Will repeat it back"
prompt += "\n ENter 'Quit' when it is done"
active = True
while active:
    message = input(prompt)
    if message == 'Quit':
        break
    else:
        print(message)
```

```
 Hi i'm Prasanth, Please tell me something
 Will repeat it back
 ENter 'Quit' when it is doneHiiiii
Hiiiii

 Hi i'm Prasanth, Please tell me something
 Will repeat it back
 ENter 'Quit' when it is donei'm P J
i'm P J

 Hi i'm Prasanth, Please tell me something
 Will repeat it back
 ENter 'Quit' when it is doneHow are you all
How are you all

 Hi i'm Prasanth, Please tell me something
 Will repeat it back
 ENter 'Quit' when it is doneQuit
```

In [38]:
```
# Directly using "While True" in the same above method:
```

In [39]:
```
prompt = "\n Hi i'm Prasanth, Please tell me something"
prompt += "\n Will repeat it back"
prompt += "\n ENter 'Quit' when it is done"

while active:
    message = input(prompt)
    if message == 'Quit':
        break
    else:
        print(message)
```

```
 Hi i'm Prasanth, Please tell me something
 Will repeat it back
 ENter 'Quit' when it is doneHi Friends How are you all
Hi Friends How are you all

 Hi i'm Prasanth, Please tell me something
 Will repeat it back
 ENter 'Quit' when it is doneQuit
```

In [45]:
```
prompt = "\n Hi, What cities you have been visited\n"
prompt += "\n ENter 'Quit' when it is done\t"

while active:
    city = input(prompt)
    if city == 'Quit':
        break
    else:
        print("\nI have been to : \n" + city + "....!")
```

```
 Hi, What cities you have been visited

 ENter 'Quit' when it is done   HYDERABAD

I have been to :
HYDERABAD ....!

 Hi, What cities you have been visited

 ENter 'Quit' when it is done   VISAKHAPATNAM

I have been to :
VISAKHAPATNAM....!

 Hi, What cities you have been visited

 ENter 'Quit' when it is done   quit

I have been to :
quit....!

 Hi, What cities you have been visited

 ENter 'Quit' when it is done   Quit
```

In [46]:
```
# Python Functions :
# page 107
# user defined function : UDF
# pre defined function : PDF
```

In [47]:
```
# Here 'greet' is a Function Name :

def greet():
    print("Hello Good Morning")
```

In [48]:
```
greet
```

Out[48]:
```
<function __main__.greet()>
```

In [49]:
```
greet()
```

```
Hello Good Morning
```

In [52]:
```
# Giving Doc String to same above sum : page 111
```

In [53]:
```
def greet():
    """This function simply greets"""
    print("Hello Good Morning")
```

In [54]:
```
print(greet.__doc__)
```

```
This function simply greets
```

In [57]:
```
def print_name(name):
    """This Function Greets with Name"""
    print("Hello...! " + str(name))
```

In [58]:
```
print_name("Suresh")
```

```
Hello...! Suresh
```

In [59]:
```
print(print_name.__doc__)
```

```
This Function Greets with Name
```

In [61]:
```
# Dual Arguments using String format :
# page 113
```

In [62]:
```
def greet2(name, message):
    """This function greets and also message"""
    print("Hello {0}, {1}".format(name,message))
```

In [64]:
```
greet2("Prasanth","Good Morning...!")
```

```
Hello Prasanth, Good Morning...!
```

In [65]:
```
greet2("king")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_15600\1376124459.py in <module>
----> 1 greet2("king")

TypeError: greet2() missing 1 required positional argument: 'message'
```

In [1]:
```
# page 114
# Default argument:
```

In [8]:
```
# here 'make' is a function
# 'action' is a parameter
# 'Nothing' is a default argument
# Here we are creating some action on particular word : eg- def make(action = 'Nothing'):

def make(action = 'Nothing'):
    return action
```

In [9]:
```
make
```

Out[9]:
```
<function __main__.make(action='Nothing')>
```

In [10]:
```
make(" i am Happy ")
```

Out[10]:
```
' i am Happy '
```

In [6]:
```
# now directly calling the 'make' function :
```

In [7]:
```
make()
```

Out[7]:
```
'Nothing'
```

In [11]:
```
# page 115
```

In [12]:
```
def greet (wishes):
    print("Hello...!" + wishes)
```

In [14]:
```
# error because, 'wishes' is not the default argument:

greet()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9616\574838690.py in <module>
      1 # error because, 'wishes' is not the default argument:
      2
----> 3 greet()

TypeError: greet() missing 1 required positional argument: 'wishes'
```

In [32]:
```
# Dual default argument:
# page 115
# i have used \n in the middle of the syntax. it's worked.
```

In [33]:
```
def greet2(name, message = "Good Morning"):
    print("Hello....{0},\n{1}".format(name,message))
```

In [34]:
```
greet2('PRASANTH', 'How are You, Have a Great Day ')
```

```
Hello....PRASANTH,
How are You, Have a Great Day
```

In [39]:
```
# if we didn't write the message, the message will defaultly generated
# as we mentioned before in 33rd Sum.
greet2('john')
```

```
Hello....john,
Good Morning
```

In [40]:
```
# Nested Function : with 'return()'
# page 116
```

In [49]:
```
# Here many_type is function name:

def many_type(x):
    if x < 0:
        return "Hello.Negative"
    else:
        return "Positive"
```

In [50]:
```
many_type(-9)
```

Out[50]: 'Hello.Negative'

In [51]:
```
def ram(x):
    if x < 0:
        return "Hello.Negative"
    else:
        return "Positive"
```

In [52]:
```
ram(-8)
```

Out[52]: 'Hello.Negative'

In [53]:
```
ram(91)
```

Out[53]: 'Positive'

In [54]:
```
# function (sum of list of values):
# Page 117
```

In [6]:
```
# Here '+=' means 'Sumation'
#'sum' is an inbuilt function of python, never call as an a variable,
# so we declare under_score before the 'sum'.
def get_sum(lst):
    """This function will do sum of objects given in list"""
    _sum = 0
    for num in lst:
        _sum += num
    return _sum
```

In [67]:
```
get_sum ([1,2,3,4])
```

Out[67]: 10

In [68]:
```
# Page 119
# function (args,kwargs)
# args --> Arguments
# kwargs --> Keyword Arguments
```

In [73]:
```
# def myfunc(a,b): it is default argument, we added:

def myfunc(a,b):
    return sum((a,b)) * 2
```

In [74]:
```
myfunc(2,3)
```

Out[74]: 10

In [75]:
```
myfunc(4)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9616\1481780178.py in <module>
----> 1 myfunc(4)

TypeError: myfunc() missing 1 required positional argument: 'b'
```

In [76]:
```
# page 120
```

In [13]:
```
def myfunc2(a,b,c = 0, d = 2, e = 5):
    return sum((a,b,c,d,e)) * 0.25
```

In [87]:
```
myfunc2(5,3,2)
```

Out[87]: 4.25

In [92]:
```
# here we have taken 5 arguments:

myfunc2(5,3,2,4,6)
```

Out[92]: 5.0

In [96]:
```
# Here at least, we need to give TWO Particular Values,
# and Maximum upto FIVE as per we mention in the sum no.85

myfunc2(5,2)
```

Out[96]: 3.5

In [98]:
```
myfunc2(5,3,2,6,8,9)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9616\330571052.py in <module>
----> 1 myfunc2(5,3,2,6,8,9)

TypeError: myfunc2() takes from 2 to 5 positional arguments but 6 were given
```

In [99]:
```
myfunc2(3,6,7,8,9)
```

Out[99]:
```
8.25
```

In [109]:
```
# 'args' --> Now Here, How many arguments we want, we can declare.
# page 121

# NOTE:
# Arguments helps us to declare 'n' number of positional arguments without any default arguments.
```

In [110]:
```
def myfunc3(*args):
    return sum(args) * 0.05
```

In [111]:
```
myfunc3(3)
```

Out[111]:
```
0.15000000000000002
```

In [112]:
```
myfunc3(3,4)
```

Out[112]:
```
0.35000000000000003
```

In [113]:
```
myfunc3(3,4,5,6,7,8,9,0)
```

Out[113]:
```
2.1
```

In [1]:
```
def func(*args):
    for i in args:
        print(i)
```

In [2]:
```
func(1,2,3,4,5)
```

```
1
2
3
4
5
```

In [11]:
```
# 66th sum's doc string:

print(get_sum.__doc__)
```

```
This function will do sum of objects given in list
```

In [14]:
```
print (myfunc2.__defaults__)
```

```
(0, 2, 5)
```

In [16]:
```
# None Because, we did'nt created doc string for this sum:

print(myfunc2.__doc__)
```

```
None
```

In [17]:
```
# page 123
# now 'text' here
```

In [22]:
```
text = "Python is a very popular programming language"
words = text.split(' ') # split is data by space,here we must give space in the middle of the empty
result = []  # some empty list we take
for idx,word in enumerate(words):
    if idx:
        result.append(word.lower())
print(result)
```

```
['is', 'a', 'very', 'popular', 'programming', 'language']
```

In [23]:
```
text = "Python is a very popular programming language"
words = text.split(' ') # split is data by space, here we must give space in the middle of the empty
result = []  # some empty list we take
for idx,word in enumerate(words):
    if idx < 4:
        result.append(word.lower())
print(result)
```

```
['python', 'is', 'a', 'very']
```

In [24]:
```
text = "Python is a very popular programming language"
words = text.split(' ') # split is data by space, here we must give space in the middle of the empty
result = []  # some empty list we take
for idx,word in enumerate(words):
    if idx > 4:
        result.append(word.lower())
print(result)
```

```
['programming', 'language']
```

In [29]:
```
text = "Python is a very popular programming language"
words = text.split(' ') # split is data by space, here we must give space in the middle of the empty
result = []  # some empty list we take
for idx,word in enumerate(words):
    if idx <= 4:
        result.append(word.lower())
print(result)# as per index - python is 0, 'is' is 1 position, 'a' is 2, 'popular' is in 4th position.
```

```
['python', 'is', 'a', 'very', 'popular']
```

In [31]:
```
# page 125
# USER FRIENDLY CALCULATOR : (+,-,*,%,/.....e.t.c) need to describe very perfectly
```

In [48]:
```
def add(a,b):
    return a+b
def sub(a,b):
    return a-b
def mul(a,b):
    return a*b
def div(a,b):
    return a/b

print("select options")
print("1.Addition")
print("2.Subtraction")
print("3.Multiplication")
print("4.Division") # first i need to read in this way

choice = int(input("Enter your choice 1/2/3/4\n"))
# once customer will give 1 or 2 may be, it will come and store in 'Buffer Memory of choice'

num1 = float(input("Enter First Number\t"))
num2 = float(input("Enter Second Number\t"))

if choice == 1:
    dt = add(num1,num2)
    print(dt)
elif choice == 2:
    dt = sub(num1,num2)
    print(dt)
elif choice == 3:
    dt = mul(num1,num2)
    print(dt)
elif choice == 4:
    dt = Div(num1,num2)
    print(dt)
else:
    print("Invalid")
```

```
select options
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice 1/2/3/4
1
Enter First Number      500
Enter Second Number     500
1000.0
```

page 128

PRE-DEFINED FUNCTION :

In [2]:
```
# ABSOLUTE NUMBER : Called as Exact Number, if we have Negative Numbers also,
# it will throw into an a 'Positive Number' :

num = -100
print(abs(num))
```

```
100
```

In [4]:
```
# it's data type in list
# Whatever we can do with this particular numbers[1,2,43], we can view it.

numbers = [1,2,43]
print(dir(numbers))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '_
_eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__
init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce_
_', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'so
rt']
```

In [5]:
```
type (numbers)
```

Out[5]:
```
list
```

In [6]:
```
len(numbers)
```

Out[6]:
```
3
```

page 129

FILTER METHOD ( ) OR FILTER FUNCTION ( )

In [7]:
```
# FILTER METHOD is also One of the pdf(pre-defined function)
# Already exists in the 'Python' we are Utilizing
```

In [8]:
```
# Now, identifying the positive number in the list :
```

In [9]:
```
def find_positive_numbers(num):
    if num > 0:
        return num
```

In [20]:
```
# Now it's printed in 'list format'. that's why -> print(list(number_list))

number_list = range(-10,10)  # from -10 to 10
print(list(number_list))
print("**********************************************************************")
```

```
[-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
**********************************************************************
```

In [24]:
```
# Now applying particular function to it :
# In the 'filter()' we applied an a parameters->
# 'find_positive_numbers(function name created previously)'
# This function name need to be applycable on the numbers of list(number_list)
# By using Pre-defined filter function - we are applying this function:


number_list = range(-10,10)  # from -10 to 10
print(list(number_list)) # Now the output should be printed in the LIST FORMAT.
print("**************************************************************************")
positive_num_lst = list(filter(find_positive_numbers,number_list))# Now we applied particular function
print(positive_num_lst)
```

```
[-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
**************************************************************************
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

page 130

is instance() : True / False :

In [25]:
```
# Here, There is a particular 'in-built PDF' Which is called "is instance function()".
# The Purpose of 'is instance' is to identify, 'Wheather the declared data is 'list data' or not'.
# It will give 'OUTPUT' in the format fo an a 'True' and 'False'
```

In [26]:
```
lst = [1,2,3,4]
print(isinstance(lst,list))
```

```
True
```

In [28]:
```
# 'False' because, 't' is an a 'Tuple() format'

t = (1,2,3,4)
print(isinstance (t, list)) # Here, Where 't' is an a list(lst).
```

```
False
```

In [ ]:
```

```