1. SINGLE CONDITION :

PAGE 10 BLUE NOTE BOOK

In [2]:
```
x = 6
if x > 5:
    print("x is greater than 5")
```

```
x is greater than 5
```

In [4]:
```
x = 7
if x > 5:
    print("x is greater than 5")
    print(x * 2)
```

```
x is greater than 5
14
```

2. DUAL CONDITION :

PAGE 11

In [5]:
```
# NOW let's see with an a 'else' statement :
```

In [6]:
```
x = 6
if x > 5:
    print("x is greater than 5")
    print(x * 2)
else:
    print("x is less than 5")
```

```
x is greater than 5
12
```

In [7]:
```
x = 4
if x > 5:
    print("x is greater than 5")
    print(x * 2)
else:
    print("x is less than 5")
```

```
x is less than 5
```

In [8]:
```
# Now let's see 'elif' statement :
```

In [9]:
```
age = 15
if age < 4:
    print("Your age is under 4")
elif age < 18:
    print("Under 18")
else:
    print("Valid")
```

```
Under 18
```

In [10]:
```
# page 12
```

In [12]:
```
x = 2
if x > 5:
    print("x is greater")
elif x == 5:
    print("x is equal to 5")
elif x < 5:
    print("x is less than 5")
else:
    print("Not valid")
```

```
x is less than 5
```

3. DUAL CONDITION IN SINGLE STATEMENT :

PAGE 11 BLUE NOTE BOOK:

In [18]:
```
x = 5
if x == 5 and type (x) is int:
    print("x is equal to 5 __ is integer")
    print(x)
elif x == 10 and type (x) is int:
    print("x is equal to 10 and integer")
    print(x)
```

```
x is equal to 5 __ is integer
5
```

4. LIST DATA :

PAGE 13

In [21]:
```
cars =["kia","bmw","mercedes","mg","toyota"]
for car in cars:              # Now it will read the car one by one
    if car == "bmw":          # Now applying the condition,,, This is called Nested loops
        print(car.upper())
    else:
        print(car.title())
```

```
Kia
BMW
Mercedes
Mg
Toyota
```

5. CONDITIONAL LOOPS:

PAGE 15

In [26]:
```
# whiel loop:

x = 1
while x <= 10:
    print(x)
    x = x + 1   # if we won't give this increment, then it will go on print  1 1 1 1 1 1 .....
```

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

In [29]:
```
cnt = 2
while cnt < 6:
    print(cnt)
    print("This is inside loop")
    cnt += 1     # Here also we gave an a increment:
```

```
2
This is inside loop
3
This is inside loop
4
This is inside loop
5
This is inside loop
```

In [32]:
```
cnt = 2
while cnt < 6:
    print(cnt)
    print("This is inside loop")
    cnt += 1     # Here also we gave an a increment:
else:
    print(cnt)
    print("This is outside loop")   # 6 < 6 = The condition false, so it is printing "This is outside loop"
```

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

6. READ DATA FROM LIST USING WHILE LOOP:

PAGE 20

In [37]:
```
# Here, 'pets' is a list and 'cat' is a string value / list item.

pets = ['cat','dog','cat','gold fish','cat','rabbit','cat']
print(pets)
print("**********************************************************")

while 'cat' in pets:
    pets.remove('cat')
    print(pets)
```

```
['cat', 'dog', 'cat', 'gold fish', 'cat', 'rabbit', 'cat']
**********************************************************
['dog', 'cat', 'gold fish', 'cat', 'rabbit', 'cat']
['dog', 'gold fish', 'cat', 'rabbit', 'cat']
['dog', 'gold fish', 'rabbit', 'cat']
['dog', 'gold fish', 'rabbit']
```

In [39]:
```
# important: page 21

mylist = [1,2,3,4,5,6,7,8,9,10]
for jelly in mylist:
    print(jelly)
```

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

In [42]:
```
# Here, i tried with '.title()'

for jelly in mylist:
    print(jelly)
    print("Hello.....")
    print("i'm prasanth".title())
```

```
1
Hello.....
I'M Prasanth
2
Hello.....
I'M Prasanth
3
Hello.....
I'M Prasanth
4
Hello.....
I'M Prasanth
5
Hello.....
I'M Prasanth
6
Hello.....
I'M Prasanth
7
Hello.....
I'M Prasanth
8
Hello.....
I'M Prasanth
9
Hello.....
I'M Prasanth
10
Hello.....
I'M Prasanth
```

In [44]:
```
# Here 'num' is a variable, 'mylist' is already defined list of 39th sum:

for num in mylist:
    if num % 2 == 0:
        print(num)
```

```
2
4
6
8
10
```

7. PRINT LETTER BY LETTER:

PAGE 23

In [49]:
```
mystring = "Hello python...."
for letter in mystring:
    print(letter)
```

```
H
e
l
l
o

p
y
t
h
o
n
.
.
.
.
```

In [50]:
```
# Here i have used '.upper()' to the above sum:

mystring = "Hello python....".upper()
for letter in mystring:
    print(letter)
```

```
H
E
L
L
O

P
Y
T
H
O
N
.
.
.
.
```

8. SETS:

MORE THAN ONE TUPLE IN A LIST IS A 'SET'. [( )] : PAGE 24

In [51]:
```
mylist
```

Out[51]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [52]:
```
# NOW giving new values to mylist:
```

In [53]:
```
mylist = [(1,2),(3,4),(5,6),(7,8)]  # list of objects:
```

In [55]:
```
# Returns No.of tuples in list:

len(mylist)
```

Out[55]: 4

In [56]:
```
for item in mylist:
    print(item)
```

```
(1, 2)
(3, 4)
(5, 6)
(7, 8)
```

In [57]:
```
# BUT NOW I WANT TO PRINT ONLY ONE SIDE OBJECTS:
```

In [58]:
```
for (a,b) in mylist:
    print(a)
```

```
1
3
5
7
```

In [60]:
```
# Also we can print both the sides in a single column:
# page 25
```

In [61]:
```
for (a,b) in mylist:
    print(a)
    print(b)
```

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

In [62]:
```
# sometimes we can take like this also:
```

In [64]:
```
mylist2 = [(1, 2,3), (4,5,6), (5,1,3)]
```

In [67]:
```
for (a,b,c) in mylist2:
    print(b)
```

```
2
5
1
```

In [68]:
```
# NOW 'SET OF KEYS' --> 'DICTIONARY KEYS':
```

In [69]:
```
d = {'k1':1, 'k2':2, 'k3':3}
```

In [70]:
```
type (d)
```

Out[70]:
```
dict
```

In [71]:
```
d
```

Out[71]:
```
{'k1': 1, 'k2': 2, 'k3': 3}
```

In [72]:
```
for item in d:
    print(item)
```

```
k1
k2
k3
```

In [74]:
```
# Whenever i declare 'd.items()', then 'd' is the 'data'. In that 'd' i want to print all the items. "Dictionary type

for item in d.items():
    print(item)
```

```
('k1', 1)
('k2', 2)
('k3', 3)
```

In [80]:
```
# page 27:

for key, value in d.items():
    print(key)
    print("**************************") # Where i'm taking differenciation.
    print(value)
```

```
k1
**************************
1
k2
**************************
2
k3
**************************
3
```

9. STRING CONCATINATING:

page 27

In [95]:
```
hello = "python"
```

In [96]:
```
hello
```

Out[96]:
```
'python'
```

In [84]:
```
# Now again i'm printing:
```

In [85]:
```
hello = "java"
```

In [86]:
```
hello
```

Out[86]:
```
'java'
```

In [87]:
```
# NOW HERE, I WANT TO PRINT BOTH 'PYTHON' AND 'JAVA':
```

In [97]:
```
hello += " java"    # We can give 'space before JAVA'
```

In [98]:
```
hello
```

Out[98]:
```
'python java'
```

10. PROMPT:

PAGE 29

In [11]:
```
prompt = "\n hello... im prasanth.  ".upper()
prompt += "tell me something i will repeate it....".title()
prompt += "\n Have a Great day"
```

In [12]:
```
prompt
```

Out[12]:
```
'\n HELLO... IM PRASANTH.  Tell Me Something I Will Repeate It....\n Have a Great day'
```

In [6]:
```
prompt = "\n Tell me Something  "
prompt += " i will repeate it...."
prompt += "\n Enter 'Quit', if logic Done"
```

In [8]:
```
prompt
```

Out[8]:
```
"\n Tell me Something   i will repeate it....\n Enter 'Quit', if logic Done"
```

In [9]:
```
message = " "
while message != 'Quit'   # Then only enter message
message = input(prompt)
```

```
  File "C:\Users\my pc\AppData\Local\Temp\ipykernel_13348\632166597.py", line 2
    while message != 'Quit'   # Then only enter message
                              ^
SyntaxError: invalid syntax
```

11 TUPLE :

PAGE 30

In [10]:
```
# PROPERTIES OF TUPLE:
# TUPLE accepts all types of Data,
# We can access objects in TUPLE Based on Index,
# We Can't 'Add' or 'Delete(del)' or 'Remove' or 'Append' objects from TUPLE.
```

In [4]:
```
tup = (25,36,963,"king","lucky")
```

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

Out[5]:
```
tuple
```

In [6]:
```
print(tup)
```

```
(25, 36, 963, 'king', 'lucky')
```

In [9]:
```
print(tup[1])
```

```
36
```

In [12]:
```
print(tup[1,2])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_26964\2169937751.py in <module>
----> 1 print(tup[1,2])

TypeError: tuple indices must be integers or slices, not tuple
```

12. LIST OF TUPLE COMBINATION :

PAGE 31

In [18]:
```
lt = [("sony",3,"f"),("tony",7,"m")]   # Means Name,year,Gender
```

In [19]:
```
len(lt)   # Now Length of list
```

Out[19]: 2

In [20]:
```
type(lt)
```

Out[20]: list

In [21]:
```
print(lt)
```

```
[('sony', 3, 'f'), ('tony', 7, 'm')]
```

In [22]:
```
print(lt[1])
```

```
('tony', 7, 'm')
```

13. TUPLE OF LIST :

PAGE 32

In [23]:
```
tu = ('ravi',25,['btech','mtech'])
```

In [25]:
```
type(tu)
```

Out[25]: tuple

In [26]:
```
len(tu)
```

Out[26]: 3

In [28]:
```
print(tu[3])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_26964\2863857931.py in <module>
----> 1 print(tu[3])

IndexError: tuple index out of range
```

In [29]:
```
print(tu[2])
```

```
['btech', 'mtech']
```

14. DICTIONARY :

PAGE 33

In [30]:
```
# 'Dictionary' is going to be represented by 'dict'
# 'Dictionary' is nothing but --> 'key : value objects'
# The pattern of 'Dictionary' represented in { flower brasses }
# 'Dictionary' may take any data type
# and We can't take data type 'directly'
```

In [35]:
```
another = {
    "man" : "bob",
    "women" : "alice",
    "other" : "tridy"
}
```

In [36]:
```
type(another)
```

Out[36]:
```
dict
```

In [37]:
```
len(another)
```

Out[37]:
```
3
```

In [38]:
```
another
```

Out[38]:
```
{'man': 'bob', 'women': 'alice', 'other': 'tridy'}
```

In [41]:
```
my_dict = { } # empty dictionary
```

In [42]:
```
my_dict
```

Out[42]:
```
{}
```

15. Adding Object to 'dict' based on 'key' :

page 35

In [44]:
```
my_dict["Day"] = "Friday"
another["Day"] = "Friday"  # here another is a 'old dictionary'
```

In [47]:
```
my_dict    # here my_dict equal to day of Friday
```

Out[47]:
```
{'Day': 'Friday'}
```

In [49]:
```
another      # Means, here we added New values('Day': 'Friday') to Old values :
```

Out[49]:
```
{'man': 'bob', 'women': 'alice', 'other': 'tridy', 'Day': 'Friday'}
```

16. Dict,Tuple, List:

page 36

In [50]:
```
# Now let's see Complex Data Types:
```

In [51]:
```
sales = {
    ("branch","p1") : [100,200,500],
    ("branch","p2") : [600,500,750],
    ("branch","p3") : [500,600,750]
}
```

In [52]:
```
sales
```

Out[52]:
```
{('branch', 'p1'): [100, 200, 500],
 ('branch', 'p2'): [600, 500, 750],
 ('branch', 'p3'): [500, 600, 750]}
```

In [53]:
```
type(sales)
```

Out[53]:
```
dict
```

In [54]:
```
len(sales)
```

Out[54]:
```
3
```

In [55]:
```
print(sales)
```

```
{('branch', 'p1'): [100, 200, 500], ('branch', 'p2'): [600, 500, 750], ('branch', 'p3'): [500, 600, 750]}
```

17. Input Method( ) :

page37

RUN TIME DECLARATION :

In [56]:
```
x = 25
```

In [57]:
```
x
```

Out[57]: 25

In [58]:
```
# Above i declared 'x' value equal to '25'
# but here i want to Build the 'LOGIC'
```

In [61]:
```
input()   # Here we are going declare in the 'CONSOLE' :
```

23

Out[61]: '23'

18. input( ), declare value, syntax... "on Runtime"

page 38

In [62]:
```
x = input()
```

34

In [63]:
```
x         # whenever we call 'x'
```

Out[63]: '34'

In [64]:
```
# Again we are Re-running :

x = input()
```

```
45
```

In [65]:
```
x
```

Out[65]: '45'

In [66]:
```
# or else we can declare like this also :
```

In [68]:
```
y = input("plz Enter 'Y' Value ")
```

```
plz Enter 'Y' Value 258
```

In [69]:
```
y
```

Out[69]: '258'

In [70]:
```
# now something different example :
```

In [71]:
```
a = 45
b = 65
c = a + b
print ( c )
```

```
110
```

In [72]:
```
# int  - only allows Numericals
```

In [80]:
```
# Here, We are declaring Values in Runtime :

a = int(input("Enter first value \n"))
b = int(input("Enter second value \n"))
print("********************")
c = a + b
print( c )
print("********************")
```

```
Enter first value
25
Enter second value
69
********************
94
********************
```

In [91]:
```
name = input("Enter Name plz \n")
age = int(input("Enter age plz \n"))
sal = int(float(input("Enter salary plz \n")))

print("**************************")
info = (name, age, sal)
print(info)
```

```
Enter Name plz
SREEKANTH
Enter age plz
25
Enter salary plz
50000
**************************
('SREEKANTH', 25, 50000)
```

19. CONTROL FLOW :

PAGE 41

In [93]:
```
# if loop
# if else
# if elif else
# while
# while else
# for :
# These are particular conditional loops :

# In this particular conditional flow, we have,
# BREAK  and  CONTINUE :
# BREAK - If condition is 'OK', Then give a 'BREAK'
# CONTINUE - It will Skip Current Conditon.
```

In [96]:
```
for number in range(1,10):
    print(number)
```

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

In [97]:
```
for number in range (1,10):
    if number == 7:
        break
    print(number)
```

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

In [99]:
```
for number in range (1,10):
    if number == 7:
        continue
    print(number)
```

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

20. NESTED LOOPS :

PAGE 44

In [102]:
```
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 loop")
```

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

In [103]:
```
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 loop")
```

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

21. EVEN NUMBERS :

PAGE 50

In [105]:
```
# now we are declaring '0 to 11 with the step function of 2'

list(range(0,11,2))
```

Out[105]:
```
[0, 2, 4, 6, 8, 10]
```

In [107]:
```
# here we did'nt get index 1,2,3,,,, like that
# for that we use 'enumerate method()' in next sum

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 [109]:
```
# Enumerate --> Helps to count the iterations :

word = 'abcde'
for item in enumerate (word):
    print(item)
```

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

22. INDEPENDENT :

PAGE 52

In [112]:
```
word = 'abcde'
for index,letter in enumerate(word):
    print(index)
    print(letter)
    print('\n')
```

```
0
a


1
b


2
c


3
d


4
e
```

23. ZIP METHOD() ON LIST :

PAGE 53

In [114]:
```
# whenever we want to mix the two particular list data, then we use this 'zip method()'

list1 = [4,5,6,7]
list2 = [10,20,30,40]

for item in zip(list1,list2):
    print(item)
```

```
(4, 10)
(5, 20)
(6, 30)
(7, 40)
```

In [119]:
```
# here i'm removing no.7 from list2:

list1 = [4,5,6]
list2 = [10,20,30,40]

for item in zip(list1,list2):
    print(item)
```

```
(4, 10)
(5, 20)
(6, 30)
```

24. SET : vvimp

PAGE 55

In [120]:
```
# SET is a another 'Complex Data Type'.
# In SET --> OBJECTS are represented in flower brasses{}, called as an a 'list objects'

# PROPERTIES OF SET :
# SET won't accept duplicates.
# SET won't allow adding, deleting, removing (or) appending objects.
# SET Can't Call Objects Based On Index.
```

In [121]:
```
dt = {1,2,1,2,4,5,7,8,1,2,4,5}
```

In [122]:
```
type(dt)
```

Out[122]:
```
set
```

In [125]:
```
# Here, more than '10' objects we declared, but we got only '6' Objects, Means 'SET' won't allow 'Duplicates'.

len(dt)
```

Out[125]:
```
6
```

In [127]:
```
print(dt)
```

```
{1, 2, 4, 5, 7, 8}
```

25. SET : Update Object based on SET only :

page 56 BLUE NOTE BOOK

In [129]:
```
# Direct Independent objects we can't add, But We can Update Object based on 'SET' Only:
```

In [130]:
```
dt.update([45,55])  # Tuple of List:
```

In [132]:
```
# Here, we added(Updated) '45' and '55' to the existing list:

dt
```

Out[132]:
```
{1, 2, 4, 5, 7, 8, 45, 55}
```

26. DISCARD : OR DELETE

PAGE 56

In [133]:
```
dt.discard(55)
```

In [134]:
```
dt
```

Out[134]:
```
{1, 2, 4, 5, 7, 8, 45}
```

27. FLAG : TRUE (or) FLASE :

PAGE 57

In [135]:
```
# We call Technically as 'FLAG'
# TRUE --> means, "ACCEPTED TO ENTER LOOP"
# FLASE --> means, "NOT ALLOWING TO ENTER LOOP"
```

In [140]:
```
prompt = "\n Tell me something"
prompt += " will repeat it back to you"
prompt += "\n Enter 'Quit' to end the program"

active = True
while active:  # all the things has been 'True', accepted.
    message = input(prompt)
    if message == 'Quit':
        active = False
    else:
        print (message)
```

```
 Tell me something will repeat it back to you
 Enter 'Quit' to end the program  Hi i'm PRASANTH
  Hi i'm PRASANTH

 Tell me something will repeat it back to you
 Enter 'Quit' to end the program  HOW ARE YOU ALL
  HOW ARE YOU ALL

 Tell me something will repeat it back to you
 Enter 'Quit' to end the program  Quit
  Quit

 Tell me something will repeat it back to you
 Enter 'Quit' to end the programquit
quit

 Tell me something will repeat it back to you
 Enter 'Quit' to end the programQuit
```

In [141]:
```
# NOW What Cities have visited, Collect list of cities :
```

In [143]:
```
prompt = "\n What cities have visited"
prompt += "\n Enter 'Quit' When you done \n"

while True:
    city = input (prompt)
    if city == 'Quit':
        break
    else:
        print(" I have been to " + city + ".....!")
```

```
 What cities have visited
 Enter 'Quit' When you done
MUMBAI
 I have been to MUMBAI.....!

 What cities have visited
 Enter 'Quit' When you done
  CHENNAI
 I have been to   CHENNAI.....!

 What cities have visited
 Enter 'Quit' When you done
  VISAKHAPATNAM
 I have been to   VISAKHAPATNAM.....!

 What cities have visited
 Enter 'Quit' When you done
HYDERABAD ALSO
 I have been to HYDERABAD ALSO.....!

 What cities have visited
 Enter 'Quit' When you done
Quit
```

28. PRINT VARIABLES BY STRING FORMAT :

PAGE 60

In [148]:
```
# Here 'f' means 'String format of' :

lang = "Python"
version = 3.8
print(f' I am learning {lang} version {version} ')
```

```
 I am learning Python version 3.8
```

In [149]:
```
lang = "Python"
version = '3.8'
print(f' I am learning {lang} version {version} ')
```

```
 I am learning Python version 3.8
```

In [150]:
```
# NOW some print statements, i want to print :
```

In [161]:
```
print('=' * 40)
print('author : John')
print('date : 01-FEB-2021')
print('=' * 40)
```

```
========================================
author : John
date : 01-FEB-2021
========================================
```

In [163]:
```
# Here, '\t' is mentioned inside the apostrophe

print('=' * 40)
print('\tauthor : John')
print('\tdate : 01-FEB-2021')
print('=' * 40)
```

```
========================================
        author : John
        date : 01-FEB-2021
========================================
```

AREA CALICULATIONS BY STRING FORMAT :

PAGE 60

In [168]:
```
# Area formula = pi * radius ** 2 (2 pi r square)

pi =  3.14
radius = 5
area = pi * radius ** 2
print(f' Area : {area:1f}')
```

```
 Area : 78.500000
```

29. FIND THE 'FILE' WITH PROPER 'EXTENSION / ENDING WITH ':' COLON

PAGE 61

In [178]:
```
# I Want to identify the 'file' in the 'folder name'
```

In [183]:
```
filename = 'civil.txt'
if filename.endswith('csv'):
    print('YES')
else:
    print('NO')
```

```
NO
```

In [185]:
```
filename = '13.emp12.csv'
if filename.endswith('csv'):
    print('YES')
else:
    print('NO')
```

```
YES
```

30. IDENTIFY (OR) COUNT NUMBER OF PASSWORD CHARACTERS :

PAGE 62

In [186]:
```
password = 'london12345'
if len(password) >= 11 :
    print('password correct')
else:
    print('password incorrect')
```

```
password correct
```

In [187]:
```
password = 'london1234'
if len(password) >= 11 :
    print('password correct')
else:
    print('password incorrect')
```

```
password incorrect
```

31. CHECK 'PASSWORD' HAS 11 CHARACTERS AND SPECIAL CHARACTERS AT '!'(NOT EQUAL TO) AVAILABLE (OR) NOT :

PAGE 63

In [188]:
```
# Here we are taking 'AND' Operator and SPECIAL CHARACTER '!' in password available or not : Means, Giving two conditi
```

In [194]:
```
password = 'london12345!'
if len(password) >= 11 and '!' in password:
    print('password correct')
else:
    print('password incorrect')
```

```
password correct
```

In [197]:
```
project_ids = ['01234','242526']
project_id = '01235'
if not project_id in project_ids:
    project_ids.append(project_id)
print(project_ids)
```

```
['01234', '242526', '01235']
```

32. Writing a LOGIC that find all '2' Digit numbers, Which divided by '11' and also indivisable by '3' : Here We are using 'LOOP' :

page 65

In [213]:
```
result = []   # Taking Empty list:

for i in range(10,100):   # Here,We are giving Range(10 to 100)
    if i % 11 == 0 and i % 3 != 0:
        result.append(str(i))
print(','.join(result))   # Here, We are giving 'Delimiter',.join
```

```
11,22,44,55,77,88
```

33. PYTHON FUNCTIONS :

PAGE 66 BLUE NOTEBOOK

In [214]:
```
# Function is a Group of related Statements that perform particular task.
# Function helps - Break our program into similar and smaller modular chunks.
# Function Avoids Repetation and Makes Code reusable.
```

TYPES OF FUNCTIONS : UDF & PDF

In [215]:
```
# UDF - User Defined Function,
# PDF - Pre Defined Function.
```

In [216]:
```
# UDF - "Based on user requirement" - Build the function (or) logic:
```

In [217]:
```
# PDF - "This will be given by Python Environment", Means, Whatever Python given, we are utilizing.
# Means, 'Pre-defined' (or) Already Defined we are Utilizing.
```

SYNTAX FOR FUNCTION :

In [218]:
```
# Function always represents - 'def function_name(parameters/passing values)':
# Function always 'ends' with an a 'colon :'
# Optional Functions - 1. We can delare an a 'Doc string', 2. We can utilize 'return'.

# 1. 'Doc-string' - It Describe/Comment/Explain about the particular Function. It is Optional.

# 2. 'return' - In Other Languages - 90% they required Mandatorily 'return' Statement.
# But in PYTHON - We don't require an a 'return' Statement. It is also an a 'Optional part' only.
```

HOW TO DECLARE THE FUNCTION :

page 73

In [220]:
```
# def(define) - Marks to Start Function-Header.
# Parameter (or) Argument - Through which We pass value to Function.
# Colon ':' - To mark the 'End' of Function-Header.
# Optionals - 1. doc_string - Describes about Function, 2. return - To 'return' a value from the Function.
```

In [221]:
```
def print_name(name):    # The Argument - i'm declaring 'name': and 'print_name' is a Function name.
    """This Function prints name"""
    print("Hello....!" + str(name))
```

In [228]:
```
print_name(' PRASANTH P J')
```

```
Hello....! PRASANTH P J
```

In [230]:
```
print_name(' KING')
```

```
Hello....! KING
```

34. 'NO ARGUMENT' FUNCTION :

PAGE 74

In [238]:
```
def print_2():   # Here, Directly i'm declaring:
    """This Function without arg::::"""
    print("This is 'No Argument Function'")
```

In [239]:
```
print_2()
```

```
This is 'No Argument Function'
```

35. READING 'doc_string' OF PARTICULAR FUNCTION :

PAGE 75

In [240]:
```
print(print_name.__doc__)
```

```
This Function prints name
```

In [243]:
```
print(print_2.__doc__)
```

```
This Function without arg::::
```

36. SET ::: COMPLEX DATA TYPES :

PAGE 76

In [244]:
```
# SET - Represents in {}Flower Brasses.
# {} - Means 'Set of Objects'.  Whatever we declare here is called as 'Set of Objects'.
```

PROPERITIES OF ' SET ' :

In [245]:
```
# A 'SET' is an a 'UNORDERED COLLECTION OF OBJECTS'
# We Can't call Objects from Set 'Based on Index'
# SET Won't Allow Duplicates
# We Can Add Objects(Single or Multiple) to SET
# SET is used to Perform for 'Mathematical Operations' such as (Union, intersection, difference e.t.c)
```

In [307]:
```
dt = {1,1,21,2,12,3,4,3,4,5,3,2,1}
```

In [308]:
```
type(dt)
```

Out[308]:
```
set
```

In [309]:
```
len(dt)     # SET Won't allow Duplicates.
```

Out[309]: 7

In [310]:
```
dt
```

Out[310]: {1, 2, 3, 4, 5, 12, 21}

In [311]:
```
str2 = {'ram','king','horse','ram','king'}
```

In [312]:
```
str2
```

Out[312]: {'horse', 'king', 'ram'}

In [313]:
```
str3 = {'ram',3,4,5,2,'king','horse',3,4,5,'ram','king'}
```

In [314]:
```
# Both Numeric and Alphabetical are applicable, But 'Numeric' Will Applicable First:

str3
```

Out[314]: {2, 3, 4, 5, 'horse', 'king', 'ram'}

ADD OBJECTS TO SET : SINGLE OBJECT

PAGE 78

In [315]:
```
# Taking 'dt' instead of 'str'
```

In [316]:
```
dt.add(556)
```

In [317]:
```
dt
```

Out[317]:
```
{1, 2, 3, 4, 5, 12, 21, 556}
```

UPDATE OBJECTS TO LIST : MULTIPLE OBJECTS

In [318]:
```
# Here, In ADD - We can Add only Single Objects.
# But, In UPDATE - We can Add Multiple Objects.

dt.update([55,66,77])
```

In [319]:
```
dt
```

Out[319]:
```
{1, 2, 3, 4, 5, 12, 21, 55, 66, 77, 556}
```

UPDATE OF 'LIST' and also 'SET' :

PAGE 78

In [320]:
```
# Here, Directly we are updating both 'LIST' and also 'SET' combinedly.
# This is the ADVANTAGE of 'SET'.

dt.update ([85,95,55,65],{25,1575,95})
```

In [321]:
```
dt
```

Out[321]:
```
{1, 2, 3, 4, 5, 12, 21, 25, 55, 65, 66, 77, 85, 95, 556, 1575}
```

DISCARD(DELETE) :

PAGE 79

In [322]:
```
dt.discard(1575)
```

In [324]:
```
dt
```

Out[324]:
```
{1, 2, 3, 4, 5, 12, 21, 25, 55, 65, 66, 77, 85, 95, 556}
```

37. THE OPERATIONS : SET

PAGE 80

UNION OPERATIONS : SET

In [349]:
```
set1 = {1,2,3,4,5,6,7}
set2 = {3,4,5,6,7,8,9}

# UNION :

print(set1 | set2)  # Here, The PIPE SYMBOL(|), Which we call as an a 'UNION'
print(set1.union(set2))  # or We can Write like this also
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

INTERSECTION : SET

PAGE 81

In [350]:
```
set1 = {1,2,3,4,5,6,7}
set2 = {3,4,5,6,7,8,9}

# INTERSECTION :

print(set1 & set2)   # '&(AMPRESENT)'
print(set1.intersection(set2)) # or We Can Write like this also.
```

```
{3, 4, 5, 6, 7}
{3, 4, 5, 6, 7}
```

DIFFERENCE : SET

PAGE 81

In [351]:
```
set1 = {1,2,3,4,5,6,7}
set2 = {3,4,5,6,7,8,9}

# DIFFERENCE :

print(set1 - set2)  # ' - DIFFERENCE'
print(set1.difference(set2))  # or We Can Write like this also.
```

```
{1, 2}
{1, 2}
```

In [352]:
```
set1 = {1,2,3,4,5,6,7}
set2 = {3,4,5,6,7,8,9}

# DIFFERENCE : REVERSE (vise versa)

print(set2 - set1)  # ' - DIFFERENCE'
print(set2.difference(set1))  # or We Can Write like this also.
```

```
{8, 9}
{8, 9}
```

38. CONDITIONAL LOOPS :

PAGE 82

In [1]:
```
# if loop - Single Condition
# if else loop - Two Conditions
# if elif else loop - More than Two Conditions

# While loop - Is used to get an a Unlimited of Iterations.
# While else -

# Also have NESTED LOOPS(if, while, for) - 'Multiple Condition loops'

# Starting 'if', Ending 'else'
# In the Middle how many Conditions are there, that many " elif's "
# Whenever the Condition Satisfies, then Only it will 'Run':
```

In [2]:
```
# CONDITIONAL LOOPS (NOTES) :

# An 'if' Statement is a PROGRAMMING.
# The 'Conditional loops' are called as 'CONTROL FLOW'.
# In 'Conditional loops' We are going to Work With 'if' loop - Single Condition
# 'if else' - Two Conditions
# 'if elif else' - More than Two Conditions.

# Here, We are going to 'Execute' a bulk of 'Logic (or) Code', if the Condition Only Satisfies.
# The 'Conditional Statements (or) Loops' helps to build - Highend Programming Logics :

# if (Condition):
# ____-> (Intendination Space (&) Intendination BLocks)
```

In [3]:
```
# Here, 'True' means 'Accepts all data to enter loop'
# 'True' means, Any data (or) condition it will take it

if True:
    print("All Accepted...!")
```

```
All Accepted...!
```

39. 'if' Condition 'True', Then Only 'Enter Loop' :

page 88

In [5]:
```
x = 5
if x < 6:
    print("Entered Loop")
```

```
Entered Loop
```

In [7]:
```
# if we run this loop, 'Nothing is getting'. Because '9' is not less than '6'.
# So, The Below Condition 'Failed'
x = 9
if x < 6:
    print("Entered Loop")
```

In [11]:
```
x = 6
if x > 5:
    print(" 'x' is Greater than '5' ")
```

```
 'x' is Greater than '5'
```

In [17]:
```
x = 6
if x > 5:
    print("'x' is Greater than '5' ")
    print(x * 2)
```

```
'x' is Greater than '5'
12
```

40. else :

page 89

In [18]:
```
x = 5
if x > 5:
    print(("'x' is Greater than '5'"))
    print(x * 2)
else:
    print("'x' is not Greater than '5'")
```

```
'x' is not Greater than '5'
```

In [19]:
```
x = 9
if x > 5:
    print(("'x' is Greater than '5'"))
    print(x * 2)
else:
    print("'x' is not Greater than '5'")
```

```
'x' is Greater than '5'
18
```

41. More than 'TWO Conditions' :

PAGE 90

In [22]:
```
# More than Two Conditions are taken into Picture :

age = 22
if age < 4:
    print("Your age is under '4'")
elif age < 18:
    print("Under 18")
else:
    print("Valid")
```

```
Valid
```

In [24]:
```
x = 5
if x > 5:
    print("'x' is Greater than '5'")
elif x == 5:
    print("'x' is equal to '5'")
else:
    print("'x' is not GREATER than '5'")
```

```
'x' is equal to '5'
```

In [25]:
```
x = 9
if x > 5:
    print("'x' is Greater than '5'")
elif x == 5:
    print("'x' is equal to '5'")
else:
    print("'x' is not GREATER than '5'")
```

```
'x' is Greater than '5'
```

In [26]:
```
x = 2
if x > 5:
    print("'x' is Greater than '5'")
elif x == 5:
    print("'x' is equal to '5'")
else:
    print("'x' is not GREATER than '5'")
```

```
'x' is not GREATER than '5'
```

42. DUAL CONDITIONS in a 'SINGLE STATEMENT' : Using 'end' Operator :

PAGE 91

In [29]:
```
x = 15
if x == 5 and type(x) is int:
    print("'x' is equal to '5', 'x' is an 'INTEGER'")
    print(x)
elif x > 10 and type (x) is int:
    print("'x' is an 'INTEGER', But it is 'x' Val")
    print(x)
```

```
'x' is an 'INTEGER', But it is 'x' Val
15
```

In [33]:
```
x = 4
if x == 5 and type(x) is int:
    print("'x' is equal to '5', 'x' is an 'INTEGER'")
    print(x)
elif x > 10 and type (x) is int:
    print("'x' is an 'INTEGER', But it is 'x' Val")
    print(x)
else:
    print(x)
```

```
4
```

In [34]:
```
x = 5
if x == 5 and type(x) is int:
    print("'x' is equal to '5', 'x' is an 'INTEGER'")
    print(x)
elif x > 10 and type (x) is int:
    print("'x' is an 'INTEGER', But it is 'x' Val")
    print(x)
```

```
'x' is equal to '5', 'x' is an 'INTEGER'
5
```

43. NESTED LOOP (for loop and if loop) :

page 92

In [36]:
```
# Here, Reading one by one the cars and cross checking the condition :

cars = ['audi','bmw','maruti','kia','toyota']
for car in cars:
    if car == 'bmw':
        print(car.upper())
    else:
        print(car.title())
```

```
Audi
BMW
Maruti
Kia
Toyota
```

44. WHILE LOOP :

PAGE 93

In [38]:
```
# While loop is used to get an a Unlimited of Iterations :
# "CONDITION is going to be Verified, and also 'Manual Increment of loop' need to be takes places :

# In 'for loop' - for every Iteration, there will be an a "Automatic increment of an a loop",
# But, In 'While loop' - "Manual Increment of loop, We need to Declare".
```

45. INFINITY LOOP : (WHILE LOOP)

PAGE 94

In [39]:
```
x = 1
while x <= 5:
    print(x)
```

```
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

46. TO OVERCOME THIS 'INFINITY LOOP', WE NEED TO 'DECLARE INCREMENT MANUALLY' :

In [42]:
```
x = 1
while x <= 5:
    print(x)
    x = x + 1    # Manual Increment Declaration: This is an a 'While loop' :
```

```
1
2
3
4
5
```

47. TAKING 'CONSTANTS' in 'While loop' : and

Adding 'INCREMENT' : ' += '

page 95

In [47]:
```
cnt = 2
while cnt < 6:
    print(cnt)
    print(" This is Inside the Loop")
    cnt += 1
```

```
2
 This is Inside the Loop
3
 This is Inside the Loop
4
 This is Inside the Loop
5
 This is Inside the Loop
```

In [48]:
```
cnt = 2
while cnt < 6:
    print(cnt)
    cnt += 1
    print(" This is Inside the Loop")
```

```
2
 This is Inside the Loop
3
 This is Inside the Loop
4
 This is Inside the Loop
5
 This is Inside the Loop
```

48. ' While else ' :

page 97

In [50]:
```
# 6 < 6 not possible : So, 'loop failed'

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
```

49. 'BREAK' and 'CONTINUE' in CONDITIONAL LOOP :

PAGE 98

BREAK :

In [55]:
```
for number in range(1,10):
    if number == 7:
        break
    print(number)
```

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

CONTINUE :

In [56]:
```
for number in range(1,10):
    if number == 7:
        continue
    print(number)
```

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

50. NESTED LOOP : 'BREAK'

it's an a Pure Nested Loop :

PAGE 99

In [58]:
```
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
```

51. NESTED LOOP : 'CONTINUE'

it's an a Pure Nested Loop :

PAGE 100

In [59]:
```
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 [ ]: