Basics of Python contd...

1. String Manipulation

In [4]:
```
#As mentioned in previous notebook, python does not support Character Data typ
e so it is a String of Length 1
a="inceptez"
print(a[2:4])   # gives the characters from 2nd index to 3rd i.e(n-1)
print(a[:3])    # gives the characters from 0nd index to 2nd i.e(n-1)
print(a[6:])    # goives the characters from 6th index to the last
```

```
ce
inc
ez
```

In [1]:
```
word="this month is april"
word.capitalize()  # gives the 1st letter of as Capital
```

Out[1]:
```
'This month is april'
```

In [2]:
```
word="this month is april"
word.title()   #gives the 1st letter of each word as capital
```

Out[2]:
```
'This Month Is April'
```

In [9]:
```
nextword="THIS IS JUPYTER"
nextword.lower()  # returns a word in Small letter
```

Out[9]:
```
'this is jupyter'
```

In [26]:
```
a=input("Enter a String:") #This opens an input box so that user can enter a v
alue
#By default input function takes a String.For inputting other date types need
 to mention it,eg:int(input("Enter a Number:"))
a.upper() # returns a word in upper case
```

```
Enter a String:test
```

Out[26]:
```
'TEST'
```

In [12]:
```
# Multi Line Strings - there are two ways for writing multi line strings
#1st Method - 2 triple Single Quotes
'''
This is python and
we are using Jupyter
Notebook
'''
#2nd Method - 2 triple double Quotes
"""
This is python and
we are using Jupyter
Notebook
"""
```

In [20]:
```
#Encoding in Python - form in which python understands i.e different bytes in
 which Python is stored
from encodings.aliases import aliases
aliases    # this gives the list of encoding types avaiable in python
# Strings are usually in UTF-8 format
```

Out[20]:
```
{'646': 'ascii',
 'ansi_x3.4_1968': 'ascii',
 'ansi_x3_4_1968': 'ascii',
 'ansi_x3.4_1986': 'ascii',
 'cp367': 'ascii',
 'csascii': 'ascii',
 'ibm367': 'ascii',
 'iso646_us': 'ascii',
 'iso_646.irv_1991': 'ascii',
 'iso_ir_6': 'ascii',
 'us': 'ascii',
 'us_ascii': 'ascii',
 'base64': 'base64_codec',
 'base_64': 'base64_codec',
 'big5_tw': 'big5',
 'csbig5': 'big5',
 'big5_hkscs': 'big5hkscs',
 'hkscs': 'big5hkscs',
 'bz2': 'bz2_codec',
 '037': 'cp037',
 'csibm037': 'cp037',
 'ebcdic_cp_ca': 'cp037',
 'ebcdic_cp_nl': 'cp037',
 'ebcdic_cp_us': 'cp037',
 'ebcdic_cp_wt': 'cp037',
 'ibm037': 'cp037',
 'ibm039': 'cp037',
 '1026': 'cp1026',
 'csibm1026': 'cp1026',
 'ibm1026': 'cp1026',
 '1125': 'cp1125',
 'ibm1125': 'cp1125',
 'cp866u': 'cp1125',
 'ruscii': 'cp1125',
 '1140': 'cp1140',
 'ibm1140': 'cp1140',
 '1250': 'cp1250',
 'windows_1250': 'cp1250',
 '1251': 'cp1251',
 'windows_1251': 'cp1251',
 '1252': 'cp1252',
 'windows_1252': 'cp1252',
 '1253': 'cp1253',
 'windows_1253': 'cp1253',
 '1254': 'cp1254',
 'windows_1254': 'cp1254',
 '1255': 'cp1255',
 'windows_1255': 'cp1255',
 '1256': 'cp1256',
 'windows_1256': 'cp1256',
 '1257': 'cp1257',
 'windows_1257': 'cp1257',
 '1258': 'cp1258',
 'windows_1258': 'cp1258',
 '273': 'cp273',
 'ibm273': 'cp273',
 'csibm273': 'cp273',
```

```
'424': 'cp424',
'csibm424': 'cp424',
'ebcdic_cp_he': 'cp424',
'ibm424': 'cp424',
'437': 'cp437',
'cspc8codepage437': 'cp437',
'ibm437': 'cp437',
'500': 'cp500',
'csibm500': 'cp500',
'ebcdic_cp_be': 'cp500',
'ebcdic_cp_ch': 'cp500',
'ibm500': 'cp500',
'775': 'cp775',
'cspc775baltic': 'cp775',
'ibm775': 'cp775',
'850': 'cp850',
'cspc850multilingual': 'cp850',
'ibm850': 'cp850',
'852': 'cp852',
'cspcp852': 'cp852',
'ibm852': 'cp852',
'855': 'cp855',
'csibm855': 'cp855',
'ibm855': 'cp855',
'857': 'cp857',
'csibm857': 'cp857',
'ibm857': 'cp857',
'858': 'cp858',
'csibm858': 'cp858',
'ibm858': 'cp858',
'860': 'cp860',
'csibm860': 'cp860',
'ibm860': 'cp860',
'861': 'cp861',
'cp_is': 'cp861',
'csibm861': 'cp861',
'ibm861': 'cp861',
'862': 'cp862',
'cspc862latinhebrew': 'cp862',
'ibm862': 'cp862',
'863': 'cp863',
'csibm863': 'cp863',
'ibm863': 'cp863',
'864': 'cp864',
'csibm864': 'cp864',
'ibm864': 'cp864',
'865': 'cp865',
'csibm865': 'cp865',
'ibm865': 'cp865',
'866': 'cp866',
'csibm866': 'cp866',
'ibm866': 'cp866',
'869': 'cp869',
'cp_gr': 'cp869',
'csibm869': 'cp869',
'ibm869': 'cp869',
'932': 'cp932',
```

```
'ms932': 'cp932',
'mskanji': 'cp932',
'ms_kanji': 'cp932',
'949': 'cp949',
'ms949': 'cp949',
'uhc': 'cp949',
'950': 'cp950',
'ms950': 'cp950',
'jisx0213': 'euc_jis_2004',
'eucjis2004': 'euc_jis_2004',
'euc_jis2004': 'euc_jis_2004',
'eucjisx0213': 'euc_jisx0213',
'eucjp': 'euc_jp',
'ujis': 'euc_jp',
'u_jis': 'euc_jp',
'euckr': 'euc_kr',
'korean': 'euc_kr',
'ksc5601': 'euc_kr',
'ks_c_5601': 'euc_kr',
'ks_c_5601_1987': 'euc_kr',
'ksx1001': 'euc_kr',
'ks_x_1001': 'euc_kr',
'gb18030_2000': 'gb18030',
'chinese': 'gb2312',
'csiso58gb231280': 'gb2312',
'euc_cn': 'gb2312',
'euccn': 'gb2312',
'eucgb2312_cn': 'gb2312',
'gb2312_1980': 'gb2312',
'gb2312_80': 'gb2312',
'iso_ir_58': 'gb2312',
'936': 'gbk',
'cp936': 'gbk',
'ms936': 'gbk',
'hex': 'hex_codec',
'roman8': 'hp_roman8',
'r8': 'hp_roman8',
'csHPRoman8': 'hp_roman8',
'hzgb': 'hz',
'hz_gb': 'hz',
'hz_gb_2312': 'hz',
'csiso2022jp': 'iso2022_jp',
'iso2022jp': 'iso2022_jp',
'iso_2022_jp': 'iso2022_jp',
'iso2022jp_1': 'iso2022_jp_1',
'iso_2022_jp_1': 'iso2022_jp_1',
'iso2022jp_2': 'iso2022_jp_2',
'iso_2022_jp_2': 'iso2022_jp_2',
'iso_2022_jp_2004': 'iso2022_jp_2004',
'iso2022jp_2004': 'iso2022_jp_2004',
'iso2022jp_3': 'iso2022_jp_3',
'iso_2022_jp_3': 'iso2022_jp_3',
'iso2022jp_ext': 'iso2022_jp_ext',
'iso_2022_jp_ext': 'iso2022_jp_ext',
'csiso2022kr': 'iso2022_kr',
'iso2022kr': 'iso2022_kr',
'iso_2022_kr': 'iso2022_kr',
```

```
'csisolatin6': 'iso8859_10',
'iso_8859_10': 'iso8859_10',
'iso_8859_10_1992': 'iso8859_10',
'iso_ir_157': 'iso8859_10',
'l6': 'iso8859_10',
'latin6': 'iso8859_10',
'thai': 'iso8859_11',
'iso_8859_11': 'iso8859_11',
'iso_8859_11_2001': 'iso8859_11',
'iso_8859_13': 'iso8859_13',
'l7': 'iso8859_13',
'latin7': 'iso8859_13',
'iso_8859_14': 'iso8859_14',
'iso_8859_14_1998': 'iso8859_14',
'iso_celtic': 'iso8859_14',
'iso_ir_199': 'iso8859_14',
'l8': 'iso8859_14',
'latin8': 'iso8859_14',
'iso_8859_15': 'iso8859_15',
'l9': 'iso8859_15',
'latin9': 'iso8859_15',
'iso_8859_16': 'iso8859_16',
'iso_8859_16_2001': 'iso8859_16',
'iso_ir_226': 'iso8859_16',
'l10': 'iso8859_16',
'latin10': 'iso8859_16',
'csisolatin2': 'iso8859_2',
'iso_8859_2': 'iso8859_2',
'iso_8859_2_1987': 'iso8859_2',
'iso_ir_101': 'iso8859_2',
'l2': 'iso8859_2',
'latin2': 'iso8859_2',
'csisolatin3': 'iso8859_3',
'iso_8859_3': 'iso8859_3',
'iso_8859_3_1988': 'iso8859_3',
'iso_ir_109': 'iso8859_3',
'l3': 'iso8859_3',
'latin3': 'iso8859_3',
'csisolatin4': 'iso8859_4',
'iso_8859_4': 'iso8859_4',
'iso_8859_4_1988': 'iso8859_4',
'iso_ir_110': 'iso8859_4',
'l4': 'iso8859_4',
'latin4': 'iso8859_4',
'csisolatincyrillic': 'iso8859_5',
'cyrillic': 'iso8859_5',
'iso_8859_5': 'iso8859_5',
'iso_8859_5_1988': 'iso8859_5',
'iso_ir_144': 'iso8859_5',
'arabic': 'iso8859_6',
'asmo_708': 'iso8859_6',
'csisolatinarabic': 'iso8859_6',
'ecma_114': 'iso8859_6',
'iso_8859_6': 'iso8859_6',
'iso_8859_6_1987': 'iso8859_6',
'iso_ir_127': 'iso8859_6',
'csisolatingreek': 'iso8859_7',
```

```
'ecma_118': 'iso8859_7',
'elot_928': 'iso8859_7',
'greek': 'iso8859_7',
'greek8': 'iso8859_7',
'iso_8859_7': 'iso8859_7',
'iso_8859_7_1987': 'iso8859_7',
'iso_ir_126': 'iso8859_7',
'csisolatinhebrew': 'iso8859_8',
'hebrew': 'iso8859_8',
'iso_8859_8': 'iso8859_8',
'iso_8859_8_1988': 'iso8859_8',
'iso_ir_138': 'iso8859_8',
'csisolatin5': 'iso8859_9',
'iso_8859_9': 'iso8859_9',
'iso_8859_9_1989': 'iso8859_9',
'iso_ir_148': 'iso8859_9',
'l5': 'iso8859_9',
'latin5': 'iso8859_9',
'cp1361': 'johab',
'ms1361': 'johab',
'cskoi8r': 'koi8_r',
'kz_1048': 'kz1048',
'rk1048': 'kz1048',
'strk1048_2002': 'kz1048',
'8859': 'latin_1',
'cp819': 'latin_1',
'csisolatin1': 'latin_1',
'ibm819': 'latin_1',
'iso8859': 'latin_1',
'iso8859_1': 'latin_1',
'iso_8859_1': 'latin_1',
'iso_8859_1_1987': 'latin_1',
'iso_ir_100': 'latin_1',
'l1': 'latin_1',
'latin': 'latin_1',
'latin1': 'latin_1',
'maccyrillic': 'mac_cyrillic',
'macgreek': 'mac_greek',
'maciceland': 'mac_iceland',
'maccentraleurope': 'mac_latin2',
'maclatin2': 'mac_latin2',
'macintosh': 'mac_roman',
'macroman': 'mac_roman',
'macturkish': 'mac_turkish',
'ansi': 'mbcs',
'dbcs': 'mbcs',
'csptcp154': 'ptcp154',
'pt154': 'ptcp154',
'cp154': 'ptcp154',
'cyrillic_asian': 'ptcp154',
'quopri': 'quopri_codec',
'quoted_printable': 'quopri_codec',
'quotedprintable': 'quopri_codec',
'rot13': 'rot_13',
'csshiftjis': 'shift_jis',
'shiftjis': 'shift_jis',
'sjis': 'shift_jis',
```

```
 's_jis': 'shift_jis',
 'shiftjis2004': 'shift_jis_2004',
 'sjis_2004': 'shift_jis_2004',
 's_jis_2004': 'shift_jis_2004',
 'shiftjisx0213': 'shift_jisx0213',
 'sjisx0213': 'shift_jisx0213',
 's_jisx0213': 'shift_jisx0213',
 'tis260': 'tactis',
 'tis620': 'tis_620',
 'tis_620_0': 'tis_620',
 'tis_620_2529_0': 'tis_620',
 'tis_620_2529_1': 'tis_620',
 'iso_ir_166': 'tis_620',
 'u16': 'utf_16',
 'utf16': 'utf_16',
 'unicodebigunmarked': 'utf_16_be',
 'utf_16be': 'utf_16_be',
 'unicodelittleunmarked': 'utf_16_le',
 'utf_16le': 'utf_16_le',
 'u32': 'utf_32',
 'utf32': 'utf_32',
 'utf_32be': 'utf_32_be',
 'utf_32le': 'utf_32_le',
 'u7': 'utf_7',
 'utf7': 'utf_7',
 'unicode_1_1_utf_7': 'utf_7',
 'u8': 'utf_8',
 'utf': 'utf_8',
 'utf8': 'utf_8',
 'utf8_ucs2': 'utf_8',
 'utf8_ucs4': 'utf_8',
 'uu': 'uu_codec',
 'zip': 'zlib_codec',
 'zlib': 'zlib_codec',
 'x_mac_japanese': 'shift_jis',
 'x_mac_korean': 'euc_kr',
 'x_mac_simp_chinese': 'gb2312',
 'x_mac_trad_chinese': 'big5'}
```

In [21]:
```
review="this is awesome"
review.encode("utf-16")  # we are forcing/changing the type of review to utf-16
```

Out[21]:
```
b'\xff\xfet\x00h\x00i\x00s\x00 \x00i\x00s\x00 \x00a\x00w\x00e\x00s\x00o\x00m\x00e\x00'
```

1.1 String Split

In [24]:
```
sample_string="Today is 1st day of April.And it is a Wednesday"
sample_string.split('.')
# this splits the string into 2 before & after '.'. The "."can be within doubl
e quotes also
```

Out[24]:
```
['Today is 1st day of April', 'And it is a Wednesday']
```

In [25]:
```
#Lets try to see how it works when there are more than 1 '.' in a sentance
sampletry_str="Today is 1st day of April.And it is a Wednesday. It feels like
 an awesome month"
sampletry_str.split(".")
#Wherever there is a ".", it gets splitted
```

Out[25]:
```
['Today is 1st day of April',
 'And it is a Wednesday',
 ' It feels like an awesome month']
```

In [28]:
```
#Lets find whether a word exists in a String
"awesome" in sampletry_str
#This returns True as the word 'awesome' exists in the string
# Lets take an example where in amazon reviews,we would need to find the occur
ance of a Specific word
```

Out[28]:
```
True
```

In [29]:
```
month1="January"
month2="February"
print("1st month is {},2nd month is {}.".format(month1,month2)) #month1 and mo
nth 2 are assigned in the {}
```

```
1st month is January,2nd month is February.
```

In [30]:
```
print("1st month is {1},2nd month is{0}.".format(month1,month2))
#We can assign values inside {} using indexes also
```

```
1st month is February,2nd month isJanuary.
```

1.2 Strip

In [33]:
```
a="          Today is 1st day of April.And it is a Wednesday. "
b=" Today is 1st day of April.And it is a Wednesday.          "
c="       Today is 1st day of April.      And it is a Wednesday.       "

print(a.lstrip())    # removes spaces in the start of a String
print(b.rstrip())    # removes spaces in the end of a String
print(c.strip())     # removes spaces in the start and end of a String & does
 not remove in the middle
```

```
Today is 1st day of April.And it is a Wednesday.
 Today is 1st day of April.And it is a Wednesday.
Today is 1st day of April.      And it is a Wednesday.
```

In [35]:
```
#By default Strip removes the white spaces, it can remove whatever we enter in
side the function
c="##########Today is 1st day of April.      And it is a Wednesday."
print(c.strip('#'))     # removes # if available at start and end
c="##########Today is 1st day of April.      And it is a Wednesday.########"
print(c.strip('#'))       # removes # if available at start and end
c="##########Today is 1st day #### of April.      And it is a Wednesday.######
##"
print(c.strip('#'))      # removes # if available at start and end but not in
 center
```

```
Today is 1st day of April.      And it is a Wednesday.
Today is 1st day of April.      And it is a Wednesday.
Today is 1st day #### of April.      And it is a Wednesday.
```

1.3 Join, split,Count & Replace

In [2]:
```
sentence="   India is the 7th largest country in the world.     It's capital i
s Delhi"
" ".join(sentence.split())    # removes the white spaces and returns the outpu
t
```

Out[2]:
```
"India is the 7th largest country in the world. It's capital is Delhi"
```

In [5]:
```
list1=sentence.split()
" ".join(list1)    # same output as above
```

Out[5]:
```
"India is the 7th largest country in the world. It's capital is Delhi"
```

In [13]:
```
"%%".join(list1)  # Inserts # in between each word
```

Out[13]:
```
"India%%is%%the%%7th%%largest%%country%%in%%the%%world.%%It's%%capital%%is%%D
elhi"
```

In [16]:
```
"".join(list1) # Does not leave any space between each other and produces a sp
acefree sentence
```

Out[16]:
```
"Indiaisthe7thlargestcountryintheworld.It'scapitalisDelhi"
```

In [17]:
```
sentence="   India is the 7th largest country in the world.     It's capital i
s Delhi"
sentence.count('c')   # Outputs the no.of occurences of word 'c'
```

Out[17]:
```
2
```

2. Loops

In [ ]:
```
#Loops are a block of statments which repeats itself until a give condition is satisfied
#Python supports the usual logical conditions from mathematics:

'''    Equals: a == b
    Not Equals: a != b
    Less than: a < b
    Less than or equal to: a <= b
    Greater than: a > b
    Greater than or equal to: a >= b '''
```

2.1 If,else,elif Statement

In [ ]:
```
'''These conditions can be used in several ways, most commonly in "if statements" and loops.
An "if statement" is written by using the if keyword.
If statements are executed only once

1) Check for a condition : Satisfied 1) Execute a statement

2) Check for a condition : Not satisfied 2) Execute the else statement  '''
```

In [20]:
```
name = input("Enter a Country name:")
if name== "China":    # : is very important for the loop to run
    print("Country name is ",name) # spaces at the start are important i.e indentation(spaces, tabs) need to be taken care
    y=100
    z=200
    print(y+z)
    print("Inside if loop")
else:
    print("Inside else loop")
```

```
Enter a Country name:China
Country name is  China
300
Inside if loop
```

In [21]:
```
#This will throw an Indentation error as there is no space at the start of Print statement below the 'if' Statment
name = input("Enter a Country name:")
if name== "China":
print("Country name is ",name)
    print("Inside if loop")
else:
    print("Inside else loop")
```

```
  File "<ipython-input-21-52a62dc5eafa>", line 3
    print("Country name is ",name) # spaces at the start are important i.e in
dentation(spaces, tabs) need to be taken care
        ^
IndentationError: expected an indented block
```

In [24]:
```
#Python uses "elif" as an else if statement
num1=int(input("1st number is"))
num2=int(input("2nd number is"))
num3=num1+num2
if(num3>50):
    print("Sum is greater than 50")   # if the 'If' is true, this gets printed
elif num3<50:     # if the above 'if' is not true, it comes to 'elif'
    print("Sum is less than 50") # exits the loop after this gets printed
else:
    print("Sum is equal to 50")
```

```
1st number is10
2nd number is20
Sum is less than 50
```

2.2 Nested if

In [5]:
```
# Nested means an object placed inside the other
# 'Nested if' means if placed inside another if
Rohit=int(input("Rohit's runs is:"))
Dhawan=int(input("Dhawan's runs is:"))
Kohli=int(input("Kohli's runs is:"))
if Rohit+Dhawan >150:
    print("Opening partnership is good")
    if(Rohit+Dhawan+Kohli) >250:
        print("India won")
    else:
        print("India Lost")
elif Rohit+Kohli >150:
    print("1st Wicket Partnership is good")
else:
    print("India Lost")
```

```
Rohit's runs is:100
Dhawan's runs is:75
Kohli's runs is:125
Opening partnership is good
India won
```

In [6]:
```
#Another Example
Rohit=int(input("Rohit's runs is:"))
Dhawan=int(input("Dhawan's runs is:"))
Kohli=int(input("Kohli's runs is:"))
if(Rohit>50):
    if(Dhawan>50):
        if(Kohli>50):
            print("India Won")
        else:
            print("Kohli's performance is not good")
    else:
            print("Dhawan's performance is not good")
else:
     print("Rohit's performance is not good")
```

```
Rohit's runs is:100
Dhawan's runs is:20
Kohli's runs is:30
Dhawan's performance is not good
```

In [ ]:
```
#Writing the condistions in a Single line
Rohit=75
Dhawan=60
Kohli=80
if(Rohit>50 and Dhawan>50 and Kohli>50):
    print("India won")
else:
    print("Top Order 3 players did not perform well")
if(Rohit<50 or Dhawan<50 or Kohli<50):
    print("Top Order 3 players did not perform well")
else:
        print("India Won")
```

2.3 Conditional Loops

In [ ]:
```
"""Conditional Loops is a way to repeate something when a certain condition is
True.
If condition is not True, it exists the loop. And if the condition is always T
rue it is
an infinite loop
"""
#While loop - It executes a block of statement until a given condition is sati
sfied
#For loop - It executes a block of statment over a certain sequence

#For and While are ENTRY CONTROLLED LOOPS
```

In [2]:
```
#while loop
n = 4
while (n > 0):  #This loop runs until n is greater than 0
    print(n)
    n = n-1
```

```
4
3
2
1
```

In [4]:
```
sample_list=[1,2,3,4,5]
for i in sample_list:
    print(i)
```

```
1
2
3
4
5
```

In [6]:
```
'''Range can be used in for loop. It will interate the number from start to en
d i.e Starting no. is included but
ending would be similar to indexes so it is n-1. Default incement would be 1
 '''
#For Printing Even numbers
for i in range(1,11):
    if(i %2 ==0):
        print(i)
```

```
2
4
6
8
10
```

In [7]:
```
#For Printing Odd numbers
for i in range(1,11,2):   #Increments the step by 2
    if(i %2 != 0):
        print(i)
```

```
1
3
5
7
9
```

In [8]:
```
#For Printing Odd numbers by giving range in reverse from abov example
for i in range(11,1,-2):   #Increments the step by -2
    if(i %2 != 0):
        print(i)
```

```
11
9
7
5
3
```

2.4 Pass, Continue & break

In [9]:
```
#Pass - it is used when we want to have an empty if loop
#Ex: if we dont know the code logic for now and want to implement it later the
n we can use this
n=0
if(n=0):
    pass
```

In [2]:
```
#Continue - continue skips the current iternation and goes to next one
for i in range(1,6):
    print(i)
    if(i ==3):
      continue
      print(i)  # this wont be printed as continue will brak the current itera
tion
        # as per our code 3 should be printed twice so it gets printed only on
ce
```

```
1
2
3
4
5
```

In [1]:
```
#Break
for i in range(1,6):
    print(i)
    if(i ==3):
      break  # runs till 3 and breaks the iteration so that 4 & 5 are not prin
ted
      print(i)
```

```
1
2
3
```

3. Functions

In [ ]:
```
#Python has the capability to use functions
'''
Functions are block of statements that we used for performing certain opertaio
ns
Functions once defined can be called any no.of times anywhere in the program
 i.e for reusability
Parameters can be passed inside a function and it can take default values as w
ell
A Function can return a value
'''
#Syantax-Functions are usually defined by def() FunctName with a :
```

In [4]:
```
#Lets define our first function
def firstFunction():
    print('inside function')

firstFunction()   # calling the above defined function
print('Outside function')
```

```
inside function
Outside function
```

In [5]:
```
#Example where we are calling the function the first
firstFunction()

def firstFunction():
    print('inside function')
```

```
inside function
```

In [7]:
```
#Pass parameters inside a function
def sampleFunction(name):
    print("Country name is :",name)

sampleFunction("India")
```

```
Country name is : India
```

In [8]:
```
#Lets see how a default value in parameter gets printed

def sampleFunction(name="Chennai"):
    print("City name is :",name)

sampleFunction("Mumbai")  #Value 'Mumbai' gets passed to the function
sampleFunction()   #Since we are not passing any value,'Chennai' gets printed
```

```
City name is : Mumbai
City name is : Chennai
```

In [15]:
```
#Lets try to print some values from list
sampleDict={'capital':"Chennai",'Tradition':"Trichy",'Industry':"Coimbatore",
'Cotton':"Tirupur"}

def testFunct(name):
    if('Cotton' in sampleDict.keys()):
        return(sampleDict.get(name))  # Key values are returned to where the f
unction was called
    else:
        print("Not known")

print("City Name is:",testFunct('Cotton'))
print("City Name is:",testFunct('Industry'))
```

```
City Name is: Tirupur
City Name is: Coimbatore
```

In [17]:
```
#Returning multiple values

def calculation(a,b):
    return a+b,a-b,a*b,b-a,a/b

print("After performing different operations of a & b:",calculation(10,5))
```

```
After performing different operations of a & b: (15, 5, 50, -5, 2.0)
```

In [20]:
```
#Recursive functions- these are the functions that call itself again and again

#Lets take the example of a factorial
def fact(num):
    if(num==1):
        return(num)
    else:
        return(num*fact(num-1))   #this function in turn calls th fact functio
n above again

fact(5)  #executes just once
```

Out[20]: 120

In [ ]:
```

```