Python Basics

- ***Python is the general purpose programming language having lots of libraries for the Machine Learning***
- it is having 33 keywords.
- single line comments - #
- multi line comments - ''' '''
- Instructions that a python interpreter can execute are called ***Statements.***
- Numbers ---> Integers,Floating point numbers,Complex numbers
- Boolean ---> True,False
- string is a sequence of unicode characters and can be indexed
- UNICODE > ASCII
- Set is unordered collection of unique items.
- Dictionary is an unordered collection of key-value pairs.
-

In [0]:
```
x = 1
print(x)
```

```
1
```

In [0]:
```
x = "Machine Learning"
print(x)
```

```
Machine Learning
```

In [0]:
```
string = "Machine Learning"
print(string)
```

```
Machine Learning
```

In [0]:
```
print(string[0])
```

```
M
```

In [0]:
```
print(string[-1])
```

```
g
```

In [0]:
```
Lambda Functionsfor i in string :
  print(i)
```

```
M
a
c
h
i
n
e

L
e
a
r
n
i
n
g
```

In [0]:
```
print(string[::-1]) #Used for reversing the string
```

```
gninraeL enihcaM
```

In [0]:
```
strng = "MADAM"
strng[::-1]
```

Out[8]:
```
'MADAM'
```

In [0]:
```
strng = "HI,Let's Learn Machine Learning."
print(strng)
```

```
HI,Let's Learn Machine Learning.
```

In [0]:
```
strng.find("Machine") # gives us the index of the first learn in our search query
```

Out[71]:
```
15
```

In [0]:
```
lst_strng = strng.split(",")
print(lst_strng)
```

```
['HI', "Let's Learn Machine Learning."]
```

In [0]:
```
strng = "HI Let's Learn Machine Learning."
print(strng)
```

```
HI Let's Learn Machine Learning.
```

In [0]:
```
lst_strng = strng.split(" ")
print(lst_strng)
```

```
['HI', "Let's", 'Learn', 'Machine', 'Learning.']
```

In [0]:
```
st = " ".join(lst_strng)
print(st)
```

```
HI Let's Learn Machine Learning.
```

In [0]:
```
strng.capitalize()   # Capitalize first letter of the string
```

Out[14]:
```
"Hi let's learn machine learning."
```

Lists

- ***Lists are ordered sequence of items need not be of same type like arrays.***

In [0]:
```
lst = [1,2,3,4,5]
lst
```

Out[15]:
```
[1, 2, 3, 4, 5]
```

In [0]:
```
lst.append(6) #adds at the end of thhe list
```

In [0]:
```
lst
```

Out[17]:
```
[1, 2, 3, 4, 5, 6]
```

In [0]:
```
lst.pop()  #Remove last element like LIFO manner in stack
lst
```

Out[18]:
```
[1, 2, 3, 4, 5]
```

In [0]:
```
lst.insert(2,2.5) # Inserts at a particular position
lst
```

Out[19]:
```
[1, 2, 2.5, 3, 4, 5]
```

In [0]:
```
lst.remove(2.5)
```

In [0]:
```
lst.sort(reverse=True)
```

In [0]:
```
lst
```

Out[24]:
```
[5, 4, 3, 2, 1]
```

In [0]:
```
lst.reverse()
```

In [0]:
```
lst
```

Out[26]:
```
[1, 2, 3, 4, 5]
```

- List sorting follows tim sort

Tuples

In [0]:
```
t=(1,'machine learning',[1,2,'ml'])
```

In [0]:
```
t
```

Out[28]:
```
(1, 'machine learning', [1, 2, 'ml'])
```

In [0]:
```
tple= 'ml'
```

In [0]:
```
type(tple)
```

Out[30]:
```
str
```

In [0]:
```
tple=('ml')
```

In [0]:
```
type(tple)
```

Out[33]:
```
str
```

In [0]:
```
tple=('ml',) #tuple single term initialisation always needs to be end with ','
```

In [0]:
```
type(tple)
```

Out[35]:
```
tuple
```

Indexing,slicing,Accessing can be done on tuple

In [0]:
```
t[1:]
```

Out[36]:
```
('machine learning', [1, 2, 'ml'])
```

In [0]:
```
t[1]
```

Out[37]:
```
'machine learning'
```

- ***Changing*** of a tuple item causes error becauseit is immutable but if item in it mutable we can change it.

In [0]:
```
t[1]='data science'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-38-276a511ead7c> in <module>()
----> 1 t[1]='data science'

TypeError: 'tuple' object does not support item assignment
```

In [0]:
```
t.index("machine learning")
```

Out[41]:
```
1
```

In [0]:
```
len(t)
```

Out[42]:
```
3
```

Sets

A set is an unordered collection of items and every element is unique(no duplicates).It is mutable

In [0]:
```
sets = {1,1,1,1,1,1,2,2,2,2,3,45}
sets
```

Out[20]:
```
{1, 2, 3, 45}
```

In [0]:
```
sets.add(2)
```

In [0]:
```
sets
```

Out[22]:
```
{1, 2, 3, 45}
```

In [0]:
```
sets.update([5,6,7])
```

In [0]:
```
sets
```

Out[24]:
```
{1, 2, 3, 5, 6, 7, 45}
```

dict,list,sets are mutable and tuple are immutable and dict and sets are unordered.

In [0]:
```
sets.discard(5)
```

In [0]:
```
sets
```

Out[26]:
```
{1, 2, 3, 6, 7, 45}
```

In [0]:
```
sets.pop()
```

Out[27]:
```
1
```

In [0]:
```
sets_ = {1,3,8,9}
```

In [0]:
```
sets_
```

Out[30]:
```
{1, 3, 8, 9}
```

In [0]:
```
sets|sets_ #union
```

Out[31]:
```
{1, 2, 3, 6, 7, 8, 9, 45}
```

In [0]:
```
sets&sets_
```

Out[32]:
```
{3}
```

In [0]:
```
sets^sets_
```

Out[33]:
```
{1, 2, 6, 7, 8, 9, 45}
```

Dictionary

- Python dictionary is an unordered collection of items following key value pair

In [0]:
```
dictn ={1:'data science',2:'ml',3:'deep learning'}
```

In [0]:
```
dictn.keys()
```

Out[35]:
```
dict_keys([1, 2, 3])
```

In [0]:
```
dictn.items()
```

Out[36]:
```
dict_items([(1, 'data science'), (2, 'ml'), (3, 'deep learning')])
```

In [0]:
```
dictn.values()
```

Out[37]:
```
dict_values(['data science', 'ml', 'deep learning'])
```

In [0]:
```
for i in dictn.items() :
  print(i[0],":",i[1])
```

```
1 : data science
2 : ml
3 : deep learning
```

Loops

In [0]:
```
print("Enter 1 to 10 numbers squares :")
i=1
while(i<=10) :
  # i*i or i**2 = squares of i
  print(i,"square is",i*i)
  i+=1
```

```
Enter 1 to 10 numbers squares :
1 square is 1
2 square is 4
3 square is 9
4 square is 16
5 square is 25
6 square is 36
7 square is 49
8 square is 64
9 square is 81
10 square is 100
```

In [0]:
```
print("Enter 1 to 10 numbers squares :")
i=1
for i in range(1,11) : # 11 because of exclusive of end
  # i*i or i**2 = squares of i
  print(i,"square is",i*i)
  i+=1
```

```
Enter 1 to 10 numbers squares :
1 square is 1
2 square is 4
3 square is 9
4 square is 16
5 square is 25
6 square is 36
7 square is 49
8 square is 64
9 square is 81
10 square is 100
```

In [0]:
```
print("Enter 1 to 10 odd numbers squares :")
i=1
for i in range(1,11,2) : # 11 because of exclusive of end and 2 indicates step_size how man
  # i*i or i**2 = squares of i
  print(i,"square is",i*i)
  i+=1
```

```
Enter 1 to 10 odd numbers squares :
1 square is 1
3 square is 9
5 square is 25
7 square is 49
9 square is 81
```

Operators

- +,-,*,/,//,%,^=** are the arthimetic operators
- in,not in,is,is not are membership operator
- ==,<,<=,>,>=,!= are comparison operators
- and,or, not are logical operators

In [0]:
```
a = 5
b= 3
```

In [0]:
```
a+b
```

Out[43]: 8

In [0]:
```
a-b
```

Out[44]: 2

In [0]:
```
a*b
```

Out[45]: 15

In [0]:
```
a/b
```

Out[46]: 1.6666666666666667

In [0]:
```
a//b # Quotient
```

Out[48]: 1

In [0]:
```
a%b # remainder
```

Out[49]: 2

In [0]:
```
a**b # 5^3
```

Out[50]: 125

In [0]:
```
a is b
```

Out[51]: False

In [0]:
```
a is not b
```

Out[52]: True

In [0]:
```
lst = [1,2,3,4,5]
```

In [0]:
```
a in lst
```

Out[54]: True

In [0]:
```
a not in lst
```

Out[55]: False

In [0]:
```
a == b
```

Out[56]: False

In [0]:
```
a!=b
```

Out[57]: True

In [0]:
```
a<b
```

Out[58]: False

In [0]:
```
a<=b
```

Out[59]: False

In [0]:
```
a>b
```

Out[60]: True

In [0]:
```
a>=b
```

Out[61]: True

In [0]:
```
c = 3
```

In [0]:
```
b == c
```

Out[64]: True

In [0]:
```
a and 0
```

Out[66]: 0

In [0]:
```
a or 0
```

Out[67]: 5

In [0]:
```
not a
```

Out[68]: False

Functions

- Function is a group of related statements that perform a specific task.It is nothing but breaking our program into smaller modular chunks.It avoids repetition and makes code reusable
- def is a keyword used to start the program
- function name is the name of the function
- parameters are input to the function
- return is the output of the program

In [0]:
```
def squaring(x) :
  """
  This program is about finding square of a given number.
  """
  return x*x
```

In [0]:
```
squaring.__doc__ #To get doc string
```

Out[120]: '\n This program is about finding square of a given number.\n '

In [0]:
```
squaring(5)
```

Out[121]: 25

In [0]:
```
squaring(-3)
```

Out[122]: 9

In [0]:
```
def positive(num) :
  """
  only positive numbers
  """
  if(num > 0) :
    return(num*num)
```

MAP,FILTER

In [0]:
```
lst = [-3,-2,-1,0,1,2,3]
```

In [0]:
```
squared = list(filter(positive,lst))
print(squared)
```

```
[1, 2, 3]
```

In [0]:
```
squared = list(map(squaring,lst))
print(squared)
```

```
[9, 4, 1, 0, 1, 4, 9]
```

Types of Functions

- ***Function arguments***
- ***Keyword Arguments***
- ***Arbitary Arguments***
- ***Recursive Functions***
- ***Lambda Functions***

In [0]:
```
def func(a,b) :
  return a+b
```

In [0]:
```
func(2,5)
```

Out[128]:
```
7
```

In [0]:
```
def func(a,b,c=1) : #  default arguments
  return a+b+c
```

In [0]:
```
func(2,5,7)
```

Out[130]:
```
14
```

In [0]:
```
func(2,5)
```

Out[131]:
```
8
```

In [3]:
```
def func(** kwargs) :
  """
  This is used to learn about keyword arguments.
  """
  if kwargs :
    print("Hello {0},{1}".format(kwargs['info'],kwargs['msg']))
func(info='ml',msg='please explain know?')
```

```
Hello ml,please explain know?
```

In [4]:
```
def func(*names) :
  """
  This function calls all persons in the names tuple
  """
  for name in names :
    print("Hello",name)
func('ml','data-science')
```

```
Hello ml
Hello data-science
```

Recursive function : Function calling itself until a condition is met.

In [0]:
```
def fib(n) :
  if(n <= 1) :
    return n
  else :
    return(fib(n-1)+fib(n-2))
```

In [22]:
```
fib(9)
```

Out[22]:
```
34
```

Lambda Function

- Lambda function is an anonymous function that is defined with out a name.
- Syntax : lambda arguments :expression

In [0]:
```
def  square(x) :
  return x*x
```

In [24]:
```
square(5)
```

Out[24]:
```
25
```

In [25]:
```
square = lambda x:x*x
square(5)
```

Out[25]:
```
25
```

- map,filter,reduce can be operated easily on lambda function

In [26]:
```
lst = [1,2,3,4,5]
print(list(map(square,lst)))
```

```
[1, 4, 9, 16, 25]
```

In [0]:
```
lst = [2,3,4,5,6,7]
```

In [31]:
```
def greater_3(n) :
  if(n>=3) :
    return n
print(list(filter(greater_3,lst)))
```

```
[3, 4, 5, 6, 7]
```

In [36]:
```
print(list(filter(lambda x: x if x>=3 else None,lst)))
```

```
[3, 4, 5, 6, 7]
```

Exception Handling

- try,except and finally are the key words used for the error handling in the python

In [0]:
```
def positive_number() :
  try :
    num = int(input("Enter the number"))
    if(num <=0) :
      raise ValueError("Error:Entered the negative number")
  except ValueError as e :
    print(e)
```

In [41]:
```
positive_number()
```

```
Enter the number-5
Error:Entered the negative number
```

In [0]:
```
def positive_number() :
  try :
    num = int(input("Enter the number"))
    if(num <=0) :
      raise ValueError("Error:Entered the negative number")
  except ValueError as e :
    print(e)
  finally :
    print("Successfully debugged the program!!")
```

In [43]:
```
positive_number()
```

```
Enter the number-10
Error:Entered the negative number
Successfully debugged the program!!
```

In [0]: