Spark Developer Training - 3 Days

Manaranjan Pradhan
manaranjan@enablecloud.com
This notebook is given as part of Spark Training to Participants. Forwarding others is strictly prohibited.

Python Basics for Spark Developers

This tutorial is meant to revise basic features of python programming for the participants. This will help participants start developing programs using Spark framework.

The topic that will be covered here are:

- Declating variables and prinitng
- Arithmatic or logical operations on variables
- Built-in functions
- Control flow statements
- Working with Data Structures: List, Tuple, Set & Dictionary
- Dealing with String
- Functions in Python
- Lambda functions
- Classes

Declaring Variables and prinintg

In [1]:

```
var1 = 2
var2 = 5
```

In [2]:

```
var1
```

Out[2]:

```
2
```

In [3]:

```
print( var1 )
```

```
2
```

In [4]:

```
mystring = 'This is python'
print( mystring )
```

```
This is python
```

In [5]:

```
print( var1, var2, mystring )
```

```
2 5 This is python
```

Operations on variables .. Arithmatic or logical

In [6]:

```
var1 + var2
```

Out[6]:

```
7
```

In [7]:

```
var1 * var2
```

Out[7]:

```
10
```

In [8]:

```
var1 == 2
```

Out[8]:

```
True
```

In [9]:

```
var1 == var2
```

Out[9]:

```
False
```

Built-in functions

In [10]:

```
round( 1.234 )
```

Out[10]:

```
1
```

In [11]:

```
# Round upto a number of decimal values
round( 1.234, 2 )
```

Out[11]:

```
1.23
```

In [12]:

```
# Importing a math function
import math
```

In [13]:

```
math.ceil( 1.2 )
```

Out[13]:

```
2
```

In [14]:

```
math.floor( 1.2 )
```

Out[14]:

```
1
```

In [15]:

```
abs( -1.2 )
```

Out[15]:

```
1.2
```

In [16]:

```
# Get the variable type
type( var1 )
```

Out[16]:

```
int
```

In [17]:

```
pow( var1 , 2 )
```

Out[17]:

```
4
```

In [18]:

```
## Generate a sequence number
numbers = range( 1, 10 )
```

In [19]:

```
numbers
```

Out[19]:

```
range(1, 10)
```

In [20]:

```
type( numbers )
```

Out[20]:

```
range
```

In [21]:

```
for i in numbers:
     print( i )
```

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

In [22]:

```
len( numbers )
```

Out[22]:

```
9
```

In [23]:

```
for i in numbers:
     print(i , end = " ")
```

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

Control Flow Statements

In [24]:

```
if var1 > 1:
    print( "Bigger" )
```

```
Bigger
```

In [25]:

```
if var1 > 5:
    print( "Bigger" )
else:
    print( "Smaller" )
```

```
Smaller
```

In [26]:

```
x = 10
y = 12
if x > y:
    print ("x>y")
elif x < y:
    print ("x<y")
else:
    print ("x=y")
```

```
x<y
```

In [27]:

```
for i in range(5):
    print (i)
```

```
0
1
2
3
4
```

In [28]:

```
i = 1
while i < 5:
    print(i)
    i = i+1
print('Bye')
```

```
1
2
3
4
Bye
```

In [29]:

```
i = 1
while i < 5:
    print(i)
    i = i+1
    if i == 4:
        break
print('Bye')
```

```
1
2
3
Bye
```

In [30]:

```
i = 1
while i < 5:
    i = i+1
    if i == 3:
        continue
    print(i)
print('Bye')
```

```
2
4
5
Bye
```

Working with Data Structures

List - Collection of elements... (Elements can repeat)

In [31]:

```

```

```
## Create an empty list
a = []
```

In [32]:

```
fruits = ['apple', 'orange', 'banana', 'papaya']
```

In [33]:

```
fruits[0]
```

Out[33]:

```
'apple'
```

In [34]:

```
## Slicing an list
fruits[1:3]
```

Out[34]:

```
['orange', 'banana']
```

In [35]:

```
## Accessing the last element
fruits[-1]
```

Out[35]:

```
'papaya'
```

In [36]:

```
# how many elements in the list
len( fruits )
```

Out[36]:

```
4
```

In [37]:

```
seasonal_fruits = ['mango', 'cherry', 'watermelon']
```

In [38]:

```
all_fruits = fruits + seasonal_fruits
```

In [39]:

```
all_fruits
```

Out[39]:

```
['apple', 'orange', 'banana', 'papaya', 'mango', 'cherry', 'watermelo
n']
```

In [40]:

```
'banana' in all_fruits
```

Out[40]:

```
True
```

In [41]:

```
'grapes' in fruits
```

Out[41]:

```
False
```

In [42]:

```
all_fruits.index( 'banana' )
```

Out[42]:

```
2
```

In [43]:

```
all_fruits.append( 'grapes' )
```

In [44]:

```
all_fruits
```

Out[44]:

```
['apple',
 'orange',
 'banana',
 'papaya',
 'mango',
 'cherry',
 'watermelon',
 'grapes']
```

In [45]:

```
a = [1,1,2,4,5,6,7]
```

In [46]:

```
a
```

Out[46]:

```
[1, 1, 2, 4, 5, 6, 7]
```

In [47]:

```
min( a )
```

Out[47]:

```
1
```

In [48]:

```
max( a )
```

Out[48]:

```
7
```

In [49]:

```
## How many times an element exists in a list
a.count( 1 )
```

Out[49]:

```
2
```

In [50]:

```
a.insert( 3, 3 )
```

In [51]:

```
a
```

Out[51]:

```
[1, 1, 2, 3, 4, 5, 6, 7]
```

In [52]:

```
a.reverse()
```

In [53]:

```
a
```

Out[53]:

```
[7, 6, 5, 4, 3, 2, 1, 1]
```

In [54]:

```
a.sort()
```

In [55]:

```
a
```

Out[55]:

```
[1, 1, 2, 3, 4, 5, 6, 7]
```

Tuples - Immutable List

In [56]:

```
tup1 = ( 1, 3, 'orange' )
```

In [57]:

```
tup1
```

Out[57]:

```
(1, 3, 'orange')
```

In [58]:

```
## It is not allowed t change the tuple elements..
tup1[1] = 'a'
```

```
-----------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-58-f67dd2a4584f> in <module>()
      1 ## It is not allowed t change the tuple elements..
----> 2 tup1[1] = 'a'

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

In [59]:

```
tupa = tuple( a )
```

In [60]:

```
tupa
```

Out[60]:

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

Set - Order list of non-repeating items

In [61]:

```
b = set( [6,1,1,2,4,5] )
```

In [62]:

```
b
```

Out[62]:

```
{1, 2, 4, 5, 6}
```

In [63]:

```
b.add( 3 )
```

In [64]:

```
b
```

Out[64]:

```
{1, 2, 3, 4, 5, 6}
```

In [65]:

```
c = set( [2,4,6,7] )
```

In [66]:

```
c.union( b )
```

Out[66]:

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

In [67]:

```
b.intersection( c )
```

Out[67]:

```
{2, 4, 6}
```

In [68]:

```
c.difference( b )
```

Out[68]:

```
{7}
```

In [69]:

```
b.remove( 3 )
```

In [70]:

```
b
```

Out[70]:

```
{1, 2, 4, 5, 6}
```

In [71]:

```
b.clear()
```

In [72]:

```
b
```

Out[72]:

```
set()
```

Iterating through the elements in list or set

In [73]:

```
for i in a:
    print( i * 2 )
```

```
2
2
4
6
8
10
12
14
```

In [74]:

```
for i in b:
    print( i )
```

Using a Dictionary

In [75]:

```
d0 = {}
d1 = dict( { 'One': 1, 'Two':2 } )
d1
```

Out[75]:

```
{'One': 1, 'Two': 2}
```

In [76]:

```
d0['One'] = 1
d0['OneTwo'] = 12
print( d0 )
```

```
{'One': 1, 'OneTwo': 12}
```

In [77]:

```
d0['One']
```

Out[77]:

```
1
```

In [78]:

```
# Join two lists and create an dictionary...
names = ['One', 'Two', 'Three', 'Four', 'Five']
numbers = [1, 2, 3, 4, 5]
```

In [79]:

```
d2 = dict( zip(names,numbers) )
```

In [80]:

```
print( d2 )
```

```
{'Two': 2, 'Four': 4, 'Three': 3, 'One': 1, 'Five': 5}
```

In [81]:

```
d2.keys()
```

Out[81]:

```
dict_keys(['Two', 'Four', 'Three', 'One', 'Five'])
```

In [82]:

```
d2.values()
```

Out[82]:

```
dict_values([2, 4, 3, 1, 5])
```

In [83]:

```
d2['six'] = 6
```

In [84]:

```
d2
```

Out[84]:

```
{'Five': 5, 'Four': 4, 'One': 1, 'Three': 3, 'Two': 2, 'six': 6}
```

In [85]:

```
# Remove an element and return it
d2.pop( 'six' )
```

Out[85]:

```
6
```

In [86]:

```
d2
```

Out[86]:

```
{'Five': 5, 'Four': 4, 'One': 1, 'Three': 3, 'Two': 2}
```

Dealing with Strings

In [87]:

```
string0 = 'python'
string1 = "Data Science"
string2 = '''This is Data science
        workshop
        using Python'''
```

In [88]:

```
print( string0, string1, string2)
```

```
python Data Science This is Data science
        workshop
        using Python
```

In [89]:

```
string2.find( "Python" )
```

Out[89]:

```
53
```

In [90]:

```
string0.capitalize()
```

Out[90]:

```
'Python'
```

In [91]:

```
string0.upper()
```

Out[91]:

```
'PYTHON'
```

In [92]:

```
len( string2 )
```

Out[92]:

```
59
```

In [93]:

```
string2.split()
```

Out[93]:

```
['This', 'is', 'Data', 'science', 'workshop', 'using', 'Python']
```

In [94]:

```
string2.replace( 'Python', 'R')
```

Out[94]:

```
'This is Data science \n        workshop\n        using R'
```

Type *Markdown* and LaTeX: α^2

Functions in Python

In [95]:

```
def addElements( a, b ):
    return a + b
```

In [96]:

```
addElements( 2, 3 )
```

Out[96]:

```
5
```

In [97]:

```
addElements( 2.3, 4.5 )
```

Out[97]:

```
6.8
```

In [98]:

```
addElements( "python", "workshop" )
```

Out[98]:

```
'pythonworkshop'
```

In [99]:

```
def addElements( a, b ):
    return a, b, a + b
```

In [100]:

```
addElements( 2, 3 )
```

Out[100]:

```
(2, 3, 5)
```

In [101]:

```
addElements( 2.3, 4.5 )
```

Out[101]:

```
(2.3, 4.5, 6.8)
```

In [102]:

```
x, y, z = addElements( 4, 5 )
```

In [103]:

```
x
```

Out[103]:

```
4
```

In [104]:

```
def addElements( a, b = 4 ):
    return a + b
```

In [105]:

```
addElements( 2 )
```

Out[105]:

```
6
```

In [106]:

```
addElements( 2, 5 )
```

Out[106]:

```
7
```

In [107]:

```
def add_n(*args):
    sum = 0
    for arg in args:
        sum = sum + arg
    return sum
```

In [108]:

```
add_n( 1, 2, 3 )
```

Out[108]:

```
6
```

In [109]:

```
add_n( 1, 2, 3, 4, 5, 6 )
```

Out[109]:

```
21
```

In [110]:

```
add_n()
```

Out[110]:

```
0
```

Lambda Functions in Python

In [111]:

```
a = lambda x: x * x
```

In [112]:

```
a( 2 )
```

Out[112]:

```
4
```

In [113]:

```
a( 2 )  *  a( 2 )
```

Out[113]:

```
16
```

In [114]:

```
mylist = [1,2,3,4,5,6,7,8,9]
```

In [115]:

```
xsquare = []

for x in mylist:
    xsquare.append( pow( x, 2 ) )

print( xsquare )
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [116]:

```
map( lambda x: pow( x, 2 ), mylist)
```

Out[116]:

```
<map at 0x7fa9cdc941d0>
```

In [117]:

```
xsquare1 = list( map( lambda x: pow( x, 2 ), mylist) )
```

In [118]:

```
print( xsquare1 )
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [119]:

```
mylist1 = [1,2,3,4,5,6,7,8,9]
```

In [120]:

```
listprods = list( map( lambda x, y: x * y, mylist, mylist1 ) )
```

In [121]:

```
listprods
```

Out[121]:

```
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [122]:

```
list( filter( lambda x : x < 5, list1 ) )
```

```
-----------------------------------------------------------------------
----
NameError                                 Traceback (most recent call l
ast)
<ipython-input-122-d2cc08ce53a1> in <module>()
----> 1 list( filter( lambda x : x < 5, list1 ) )

NameError: name 'list1' is not defined
```

Classes and Objects

In [123]:

```
class Student:
    workshop = 'python'
    def __init__(self,name,age):
        self.name = name
        self.age = age
    def describe( self ):
        print( self.name, " is ", self.age, " years old and participating in ", Stude
        return
```

In [125]:

```
student1 = Student( "manaranjan", 39 )
```

In [126]:

```
student1.name
```

Out[126]:

```
'manaranjan'
```

In [127]:

```
student1.describe()
```

```
manaranjan  is  39  years old and participating in  python  class
```

In [130]:

```
Student.workshop = "Spark Developer Training"
```

In [131]:

```
student1.workshop
```

Out[131]:

```
'Spark Developer Training'
```