In [1]:
```
#CITATION :- DATA SCIENCE FROM SCRATCH CHAPTER 2 EXAMPLES
#Functions
def double(x):
    return x*2
```

In [2]:
```
double(2)
def apply_to_one(f):
    return f(1)
```

In [3]:
```
# anonymous functions
y =  apply_to_one(lambda x : x *2)
y
```

Out[3]:
```
2
```

In [4]:
```
def substract(a=0, b=0):
    return a - b
```

In [5]:
```
substract(10, 5)
```

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

In [6]:
```
#Exceptions
try:
    open("data.txt")
except:
    print ("Could not open file")
```

```
Could not open file
```

In [7]:
```
integer_list = [1,2,3]
```

In [8]:
```
x= range(10)
```

In [9]:
```
x
```

Out[9]:
```
range(0, 10)
```

In [10]:
```
#Slicing
x[:3]
```

Out[10]:
```
range(0, 3)
```

In [11]:
```
#negative slicing
x[:-3]
```

Out[11]:
```
range(0, 7)
```

In [12]:
```
x = [1,2,3]
```

In [13]:

```
x.extend([2,3,4])
```

In [14]:
```
x
```

Out[14]:
```
[1, 2, 3, 2, 3, 4]
```

In [15]:
```
#Tuples
def findSumAndProduct(x,y):
    return (x+y),(x*y)
```

In [16]:
```
findSumAndProduct(5,4)
```

Out[16]:
```
(9, 20)
```

In [17]:
```
#Dictionaries
dictionary = dict()
dictionary["abhi"] = 80
dictionary["manu"] = 95
```

In [18]:
```
dictionary.get("abhi",0)
dictionary.get("ma",0)
```

Out[18]:
```
0
```

In [19]:
```
tweet = {
"user" : "joelgrus",
"text" : "Data Science is Awesome",
"retweet_count" : 100,
"hashtags" : ["#data", "#science", "#datascience", "#awesome", "#yolo"]
}
```

In [20]:
```
key = tweet.keys()
key
```

Out[20]:
```
dict_keys(['text', 'retweet_count', 'hashtags', 'user'])
```

In [21]:
```
value = tweet.values()
value
```

Out[21]:
```
dict_values(['Data Science is Awesome', 100, ['#data', '#science', '#data
science', '#awesome', '#yolo'], 'joelgrus'])
```

In [22]:
```
items = tweet.items()
items
```

Out[22]:
```
dict_items([('text', 'Data Science is Awesome'), ('retweet_count', 100),
 ('hashtags', ['#data', '#science', '#datascience', '#awesome', '#yol
o']), ('user', 'joelgrus')])
```

```
In [23]: # default dict
         from collections import defaultdict

         words = defaultdict()
```

```
In [24]: #Counter
         from collections import Counter
```

```
In [25]: c = Counter([0, 1, 2, 0])
         c

Out[25]: Counter({0: 2, 1: 1, 2: 1})
```

```
In [26]: #sets
         s = set()
```

```
In [27]: s.add(1)
         s.add(1)
         s

Out[27]: {1}
```

```
In [28]: #sorting
         y = sorted(x)
         y

Out[28]: [1, 2, 2, 3, 3, 4]
```

```
In [29]: x.sort()
         x

Out[29]: [1, 2, 2, 3, 3, 4]
```

```
In [30]: #List Comprehensions
         square = [x*x for x in range(5)]
         square

Out[30]: [0, 1, 4, 9, 16]
```

In [31]:
```
pairs = [(x,y) for x in range(10)
        for y in range(20)]
pairs
```

Out[31]:
```
[(0, 0),
 (0, 1),
 (0, 2),
 (0, 3),
 (0, 4),
 (0, 5),
 (0, 6),
 (0, 7),
 (0, 8),
 (0, 9),
 (0, 10),
 (0, 11),
 (0, 12),
 (0, 13),
 (0, 14),
 (0, 15),
 (0, 16),
 (0, 17),
 (0, 18),
```

In [32]:
```
#generators and Iterators
def simple_range(n):
    i = 0
    while i < n:
        yield i
        i += 1
simple_range(10)
```

Out[32]:
```
<generator object simple_range at 0x104134c50>
```

In [33]:
```
#Randomness
import random
```

In [34]:
```
random.seed(10)
print (random.random())
```

```
0.5714025946899135
```

In [35]:
```
random.randrange(1,10)
```

Out[35]:
```
7
```

In [36]:
```
four_with_replacement = [random.choice(range(10))
                         for _ in range(4)]
```

In [37]:
```
four_with_replacement
```

Out[37]:
```
[7, 9, 0, 3]
```

```
In [38]: #Object Oriented Programming
         class Set:
             def __init__(self, values=None):
                 self.dict = {}
                 if values is not None:
                     for value in values:
                         self.add(value)

             def add(self, value):
                 self.dict[value] = True

         s = Set([1,2,3])
         s.add(4)
```

```
In [39]: s
```

```
Out[39]: <__main__.Set at 0x10416a8d0>
```

```
In [40]: print(s)
```

```
         <__main__.Set object at 0x10416a8d0>
```

```
In [41]: def multiply(x, y): return x * y
         products = map(multiply, [1, 2], [4, 5])
         products
```

```
Out[41]: <map at 0x104148d30>
```

```
In [42]: # zip and argument unpacking
         list1 = ['a', 'b', 'c']
         list2 = [1, 2, 3]
         zip(list1, list2)
```

```
Out[42]: <zip at 0x10417cf88>
```

```
In [43]: #args and kwargs
         def doubler(f):
             def g(x):
                 return 2 * f(x)
             return g

         def f1(x):
             return x + 1

         g = doubler(f1)

         print (g(10))
         print (g(-10))
```

```
         22
         -18
```

```
In [ ]:
```