Problem Set #2

Saharnaz Babaei Balderlou
saharnaz.babaei@grad.moore.sc.edu

University of South Carolina — September 9, 2019

Part 1 - Standard Library

Problem 1

```
>>> # Problem 1
...
>>> def myfunc(L):
...     minimum = min(L)
...     maximum = max(L)
...     avg = sum(L)/len(L)
...     # or (for average)
...     '''
...     sum = 0
...     count = 0
...     for v in L:
...         count = count + 1
...         sum = sum + v
...     avg = sum/count
...     '''
...     Lstat = [minimum, maximum, avg]
...     print(Lstat)

>>> # Example 1
...
>>> L = [1, 2, 3]
>>> myfunc(L)
[1, 3, 2.0]
>>> # Example 2: implementing function in one line
...
>>> myfunc([0, 1, 2])
[0, 2, 1.0]
```

Problem 2

```
>>> # Problem 2
...
>>> x_1 = int(2.3)      #create an integer.
>>> type(x_1)           #type of object is integer.
<class 'int'>
>>> x_2 = x_1           #create a copy.
>>> x_2 = "Saharnaz"    #change the new (copied) object.
>>> x_2 == x_1          #compare the two objects
False                   #Only new one changed
>>> type(x_1)
<class 'int'>
>>> type(x_2)
<class 'str'>
>>> #int is immutable
>>>
>>>
>>> y_1 = str(1989)   #create a string
>>> type(y_1)
<class 'str'>
```

```
>>> y_2 = y_1          #create a copy of the object
>>> y_2 = 1989      #change new object
>>> y_1 == y_2         #test for change
False
>>> type(y_2)
<class 'int'>
>>> print(y_1, y_2)
1989 1989          #Only one changed (looks same but one object is integer, the other
     one is string)
# str is immutable
>>>
>>> list1 = list([1, 2, 3]) #create a list
>>> list2 = list1       #new copy of list
>>> list2[1] = 'a'      #change new list
>>> print([[list1], [list2]]) #Both changed!
[[1, 'a', 3], [1, 'a', 3]]
>>> # list is mutable
>>>
>>> tuple1 = ('Jonathan', 'Milton', 'Leili') #create new tuple
>>> tuple2 = tuple1     #new name for tuple
>>> tuple2 += ('Saharnaz',) #change new tuple
>>> tuple1 == tuple2    #compare tuples
False                   #Only one tuple changed
>>> print(tuple1, tuple2)
(('Jonathan', 'Milton', 'Leili') ('Jonathan', 'Milton', 'Leili', 'Saharnaz'))
>>> # tuple is immutable
>>>
>>> set1 = {'alpha', 'alpha', 'a', 10} #create new set
>>> set2 = set1         #new name for the set
>>> set2 -={'alpha'}    #change the new set
>>> print(set1, set2)   #Both changed
{10, 'a'} {10, 'a'}
>>> #set is muttable
```

Problem 3

```
>>> # Problem 3
>>> import calculator as calc
>>> def pythagorean(i, j):
...     print(calc.sqrt(calc.funcS(calc.funcP(i, i), calc.funcP(j,j))))
>>>
>>> i = 3
>>> j = 4
>>> pythagorean(i, j)
5.0  # length of hypotenuse
```

Part 2 - Introductory to Numpy

Problem 1

```
>>> # Problem 1
>>> import numpy as np
>>> def mathprod(A, B):
...     R1 = np.dot(np.array(A), np.array(B))
...     print(R1)
>>>
>>> A = [[3, -1, 4], [1, 5, -9]]
>>> B = [[2, 6, -5, 3], [5, -8, 9, 7], [9, -3, -2, -3]]
>>> mathprod(A, B)
[[ 37  14 -32 -10]
 [-54  -7  58  65]]
```

Problem 2

```
>>> # Problem 2
>>> import numpy as np
>>> def func(A):
...     A = np.array(A)
...     print((-A * A * A) + (9 * A * A) - (15 * A))
>>> A = [[3, 1, 4], [1, 5, 9], [-5, 3, 1]]
>>> func(A)
array([[   9,   -7,   20],
       [  -7,   25, -135],
       [ 425,    9,   -7]])
```

Problem 5

```
>>> # Problem 5
>>> import numpy as np
>>> def blockmat(A, B, C):
...     A = np.array(A)
...     B = np.array(B)
...     C = np.array(C)
...     print(np.vstack(((np.column_stack((np.zeros((3,3)), A.T, np.eye((3))))), (np.
    column_stack((A, np.zeros((2, 2)), np.zeros((2, 3))))), (np.column_stack((B, np.
    zeros((3,2)), C))))))
>>> A = [[0, 2, 4], [1, 3, 5]]
>>> B = [[3, 0, 0], [3, 3, 0], [3, 3, 3]]
>>> C = [[-2, 0, 0], [0, -2, 0], [0, 0, -2]]
>>> blockmat(A, B, C)
[[ 0.  0.  0.  0.  1.  1.  0.  0.]
 [ 0.  0.  0.  2.  3.  0.  1.  0.]
 [ 0.  0.  0.  4.  5.  0.  0.  1.]
 [ 0.  2.  4.  0.  0.  0.  0.  0.]
 [ 1.  3.  5.  0.  0.  0.  0.  0.]
 [ 3.  0.  0.  0.  0. -2.  0.  0.]
 [ 3.  3.  0.  0.  0.  0. -2.  0.]
 [ 3.  3.  3.  0.  0.  0.  0. -2.]]
```

Part 3 - Object-Oriented Programming

Problem 1

```
>>> # Problem 1
>>> class Backpack(object):    #Part 1. constructor modification
...     def __init__(self, name, color, max_size = 5):
...         """
...         A Backpack object class. includes a name, a list of contents, a color,
    and maximum size for contents.
...
...         Attributes:
...             name (str) : the name of the backpack's owner.
...             contents (list) : the contents of the backpack.
...             color (str) : color of the backpack.
...             max_size (int) : the maximum size of the backpack.
...         """
...         self.name = name
...         self.contents = []
...         self.color = color
...         self.max_size = max_size
...     def put(self, item):    #modification of put() method
...         if len(self.contents) >= self.max_size:
...             print("No Room!")
...         else:
...             self.contents.append(item)
...
```

```
...     def dump(self):    #reset contents to an empty list
...         self.contents = []
>>>
>>> #-------------------------------------------------------------------------
>>> #-------------------------------------------------------------------------
>>> test = Backpack("Saharnaz", "Gray")
>>> for item in ["laptop", "mouse", "book", "pen", "pencil", "keys"]:
>>>     test.put(item)
>>> print("contents: ", test.contents)
No Room!
contents:  ['laptop', 'mouse', 'book', 'pen', 'pencil']
>>> #-------------------------------------------------------------------------
>>> #-------------------------------------------------------------------------
>>> test.dump()
>>> print("contents: ", test.contents)
contents:  []
```

Problem 2

```
>>> # Problem 2
>>> class Backpack(object):
...     '''
...     A Backpack object class.
...     Attributes and details in problem 1
...     '''
...     def __init__(self, name, color, max_size = 5):
...         self.name = name
...         self.color = color
...         self.max_size = max_size
...         self.contents = []
...
...     def put(self, item):    #modification of put() method
...         if len(self.contents) >= self.max_size:
...             print("No Room!")
...         else:
...             self.contents.append(item)
...
...     def dump(self):    #reset contents to an empty list
...         self.contents = []
...
>>>
>>> #-------------------------------------------------------------------------
>>> #-------------------------------------------------------------------------
>>> class Jetpack(Backpack):
...     """
...     A Jetpack object class. Inherited from Backpack class.
...     A Jetpack is smaller than a Backpackself.
...
...     Attributes:
...         name (str): the name of the Jetpack's owner.
...         color (str): color of Jetpack
...         max_size (int): maximum number of items that can be fit inside the
    Jetpack
...         fuel (int):
...         contents (list): the contents of the Jetpack
...     """
...     def __init__(self, name, color, max_size = 2, fuel =10):
...         Backpack.__init__(self, name, color, max_size)
...         self.fuel = fuel
...
...     def fly(self, burned_fuel):
...         self.burned_fuel = burned_fuel
...         if self.burned_fuel <= self.fuel:
...             self.fuel = self.fuel - self.burned_fuel
...         if self.burned_fuel > self.fuel:
...             self.fuel = self.fuel
...             print("Not Enough Fuel!")
...     def dump(self):
...         Backpack.dump(self)
...         self.fuel = 0
```

```
...
>>> #-------------------------------------------------------------------------
>>> #-------------------------------------------------------------------------
>>> test = Jetpack("Saharnaz", "White")
>>> test.put("calculator")
>>> print(test.contents, test.fuel)
['calculator'] 10
>>> test.dump()
>>> print(test.contents, test.fuel)
[] 0
```