Midterm.Python.BS.DataFrames

October 29, 2018

1 Q1 Python Functions (2 pts)

- rewrite the function foo so that it returns a tuple of the parameter, its square and its cube
- example: print (foo(2)) => (2,4,8)

```
In [1]: # rewrite foo and display tuple
        def foo(a):
            # modify
            return a
```

2 Q2 : Apply functions to a string (3 pts)

- modify the function rep1 that replaces any substring of the form 'aaa' by the string 'zzz'
- modify the function rep2 that replaces the second character of every string with 'X'
- create a list of function objects fo1 = [rep1, rep2]
- create a list of strings strList = ['welcome to aaa land', 'time to go to aaa']
- modify the function: modify(funclist, mystrlist) - that returns a list of strings after modifications
- expected output: ['wXlcome to zzz land', 'tXme to go to zzz']

```
In [2]: # code for Q2
        strList = ['welcome to aaa land', 'time to go to aaa']

        def rep1():
            pass

        def rep2():
            pass

        def modify(funclist, mystrlist):
            pass
```

3 Q3 : Usage of *args (3 pts)

- Modify the function fooargs that takes *args paramters and returns a tuple of all the parameters in ascending sorted order

- Example: fooargs('a','x','b')
- returns a TUPLE ('a','b','x')

```
In [3]: def fooargs(*argv):
            pass

        print ( fooargs('a','x', 'b'))
```

```
None
```

4 Q4 List comprehension (3 pts)

- use the range function to create a list of integers from 0 .. 9
- write a list comprehension that returns a list with 'Even' or 'Odd' depending on the value
- expect: ['Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd']

```
In [4]: vals = 'use range function here'
        resultList = ['list comprehension code for Even and Odd' ]
        print('result = ', resultList)
```

```
result =  ['list comprehension code for Even and Odd']
```

5 Q5 lambda - anonymous functions (3 pts)

- rewrite the function foo1 as a lambda

```
In [5]: #write equivalent lambda function - call it foolambda
        def foo1(x, y, z):
            return x + y /z

        # Call the function
        z = foo1(2, 3, 2)

        print (z)  # prints 3.5

        # convert the above function into a lambda function
        # here lambda returns a function object which we assign to a variable
        foolambda = 'need lambda here'

        print ( foolambda(2, 3, 2)  )  # expect 3.5 - NOT the TypeError shown below
```

```
3.5
```

```
        ---------------------------------------------------------------------
```

```
TypeError                                 Traceback (most recent call last)

<ipython-input-5-c19250f6bf19> in <module>()
 12 foolambda = 'need lambda here'
 13
---> 14 print ( foolambda(2, 3, 2)  )  # expect 3.5 - NOT the TypeError shown below


TypeError: 'str' object is not callable
```

6 Q6 filter function with lambda (3 pts)

- given mylist = [10, 20, 30, 40, 50, 60, 70]
- write code for mylambda and use it in a filter function such that only values between 40 and 60 (inclusive) are returned and displayed as a list
- expected printed output: [40,50,60]

```
In [6]: # filter with lambda
        mylist = [10, 20, 30, 40, 50, 60, 70]
        mylambda = 'replace with code'
```

7 Q7 Create Your Own Iterator in Python (4 pts)

- Refactor the class MyIter with a constructor that takes three parameters: start, end and interval.
- The class be iterable and capable of returning an iterator. The iterator should return numbers starting at 'start', ending at 'end' and with an interval of 'interval'
- For example when passing in the parameters (10,10,10) the iterator should return 10,20,30,...,100

```
In [7]: class MyIter:
            def __init__(self, start=100, end=200, interval=10 ):
                pass

            'write additional needed code here'

        #create instance of MyIter
        c1 = MyIter(10,30,10)

        # write a loop that uses iterator from c1 to print the values 10..100
```

8 Q8: Statistics : correlation (3 pts)

- start with a list: list0 = [2,4,6,8,10,12,14]
- write 3 lists: list1, list2 and list3 such that

- list0 and list1 have a correlation of +1
- list0 and list2 have a correlation of -1
- list0 and list3 have a corrleation between -0.5 and +0.5

In [8]:
```
# do: import the pearsonr function
# write code that calls the pearsonr function for the above lists


list0 = [2,4,6,8,10,12,14]
list1 = []
list2 = []
list3 = []

# show pearsonr for list0 vs list1
print ("list0 vs list1")

# show pearsonr for list0 vs list2
print ("list0 vs list2")

# show pearsonr for list0 vs list3
print ("list0 vs list3")
```

```
list0 vs list1
list0 vs list2
list0 vs list3
```

9 Q9. Beautiful Soup (7 pts)

Using Jupyter, load the website: Nasdaq XXX where you will find a list of dividend payouts and a variety of dates for Nvidia - A) Extract all the dividends and the corresponding XXX date listed on the website. - B) Load the data into a DataFrame with two columns: Date and Dividend sorted by ascending date and display the DataFrame - C) Plot the dividends over time and generate the following graph. See the graph on the MIdterm Exam handout.

10 Q10 Panda Exercise 1 (4pts)

- see the Main Midterm Handout for problem description

In [ ]:
```
# answer for Q10
```

11 Q11 Panda Exercise 2 (2 pts)

- see Main Midterm handout

In [ ]:
```
# answer for  Q11
```

12 Q12 Panda Exercise 3 (4 pts)

- see Main Midterm handout

```
In [ ]: #answer for Q12
```

13 Q13 Olympics 1 (3 pts)

- Download the Olympics.csv
- Display the count of the number of Hungarians (HUN) who won medals between 2000 and 2008 in the Discipline of Swimming or Fencing

```
In [ ]: # Answer for Q13
```

14 Q14 Olympics 2 (3pts)

- Based on Olympics.csv, in which events did Jesse Owens win medals - report as a DataFrame with column headings of: Year, City and Event

```
In [9]: # Answer for Q14
```