Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : List, Tuple and Dictionary

Name: Phua Yan Han

IC Number: 050824070059

Date : 24/6/23

Introduction : List tuple and dictionary for python

Conclusion : Learn python syntax for List, tuple and dictionary and how to scan data

EXERCISE 4

List, Tuple and Dictionary

In [ ]:
```
Note : Please start your jupyter notebook using the anaconda prompt with this c
Data Rate Exceeded Problem
At the anaconda prompt, type : jupyter notebook --NotebookApp.iopub_data_rate_l
```

Question 1

Expected answer:

In [3]:
```
# A. match_ends
# Given a list of strings, return the count of the number of
# strings where the string length is 2 or more and the first
# and last chars of the string are the same.
# Note: python does not have a ++ operator, but += works.

text1 = (['aba', 'xyz', 'aa', 'x', 'bbb']) #3
text2 = (['', 'x', 'xy', 'xyx', 'xx']) #2
text3 = (['aaa', 'be', 'abc', 'hello']) #1

def match_ends(words):
    # your code here
    return len(list(filter( lambda x:x!="" and x[0]==x[-1] and len(x)>=2 ,words



print('match_ends')
print(match_ends(text1))
print(match_ends(text2))
print(match_ends(text3))
```

```
match_ends
3
2
1
```

Question 2

Expected answer:

```
front_x
['xaa', 'xzz', 'axx', 'bbb', 'ccc']
['xaa', 'xcc', 'aaa', 'bbb', 'ccc']
['xanadu', 'xyz', 'aardvark', 'apple', 'mix']
```

In [8]:
```
# B. front_x
# Given a list of strings, return a list with the strings
# in sorted order, except group all the strings that begin with 'x' first.
# e.g. ['mix', 'xyz', 'apple', 'xanadu', 'aardvark'] yields
# ['xanadu', 'xyz', 'aardvark', 'apple', 'mix']
# Hint: this can be done by making 2 lists and sorting each of them
# before combining them.

# ['xaa', 'xzz', 'axx', 'bbb', 'ccc']
text1 = (['bbb', 'ccc', 'axx', 'xzz', 'xaa'])

# ['xaa', 'xcc', 'aaa', 'bbb', 'ccc']
text2 = (['ccc', 'bbb', 'aaa', 'xcc', 'xaa'])

# ['xanadu', 'xyz', 'aardvark', 'apple', 'mix']
text3 = (['mix', 'xyz', 'apple', 'xanadu', 'aardvark'])

def front_x(words):
    # your code here
    listX=[ i for i in words if i[0]=="x"]
    listY=[i for i in words if i[0]!="x"]
    listY.sort()
    listX.sort()
    return listX + listY

print()
print('front_x')

print(front_x(text1))
print(front_x(text2))
print(front_x(text3))
```

```

front_x
['xaa', 'xzz', 'axx', 'bbb', 'ccc']
['xaa', 'xcc', 'aaa', 'bbb', 'ccc']
['xanadu', 'xyz', 'aardvark', 'apple', 'mix']
```

Question 3

Expected answer:

```
[(2, 1), (3, 2), (1, 3)]
[(3, 1), (1, 2), (2, 3)]
[(2, 2), (1, 3), (3, 4, 5), (1, 7)]
```

In [11]:
```
# C. sort_last
# Given a list of non-empty tuples, return a list sorted in increasing
# order by the last element in each tuple.
# e.g. [(1, 7), (1, 3), (3, 4, 5), (2, 2)] yields
# [(2, 2), (1, 3), (3, 4, 5), (1, 7)]
# Hint: use a custom key= function to extract the last element form each tuple.

#output: [(2, 1), (3, 2), (1, 3)]
list1 = [(1, 3), (3, 2), (2, 1)]

#output: [(3, 1), (1, 2), (2, 3)]
list2 = [(2, 3), (1, 2), (3, 1)]

#output: [(2, 2), (1, 3), (3, 4, 5), (1, 7)]
list3 = [(1, 7), (1, 3), (3, 4, 5), (2, 2)]

def sort_last(tuples):
    # your code here
    return sorted(tuples, key=lambda x: x[-1])


print(sort_last(list1))
print(sort_last(list2))
print(sort_last(list3))
```

```
[(2, 1), (3, 2), (1, 3)]
[(3, 1), (1, 2), (2, 3)]
[(2, 2), (1, 3), (3, 4, 5), (1, 7)]
```

Question 4

In [5]:
```
# read the stocks.json file and store into 'records'
import json

%pwd

path ='./Data files/stocks.json'
stock_data = []
with open(path) as f:
    for line in f:
        stock_data.append(json.loads(line))

print(stock_data)
record = stock_data
print(record)
```

```
[{'_id': {'$oid': '52853800bb1177ca391c17ff'}, 'Ticker': 'A', 'Profit Margi
n': 0.137, 'Institutional Ownership': 0.847, 'EPS growth past 5 years': 0.1
58, 'Total Debt/Equity': 0.56, 'Current Ratio': 3, 'Return on Assets': 0.08
9, 'Sector': 'Healthcare', 'P/S': 2.54, 'Change from Open': -0.0148, 'Perfo
rmance (YTD)': 0.2605, 'Performance (Week)': 0.0031, 'Quick Ratio': 2.3, 'I
nsider Transactions': -0.1352, 'P/B': 3.63, 'EPS growth quarter over quarte
r': -0.29, 'Payout Ratio': 0.162, 'Performance (Quarter)': 0.0928, 'Forward
P/E': 16.11, 'P/E': 19.1, '200-Day Simple Moving Average': 0.1062, 'Shares
Outstanding': 339, 'Earnings Date': {'$date': 1384464600000}, '52-Week Hig
h': -0.0544, 'P/Cash': 7.45, 'Change': -0.0148, 'Analyst Recom': 1.6, 'Vola
tility (Week)': 0.0177, 'Country': 'USA', 'Return on Equity': 0.182, '50-Da
y Low': 0.0728, 'Price': 50.44, '50-Day High': -0.0544, 'Return on Investme
nt': 0.163, 'Shares Float': 330.21, 'Dividend Yield': 0.0094, 'EPS growth n
ext 5 years': 0.0843, 'Industry': 'Medical Laboratories & Research', 'Bet
a': 1.5, 'Sales growth quarter over quarter': -0.041, 'Operating Margin':
0.187, 'EPS (ttm)': 2.68, 'PEG': 2.27, 'Float Short': 0.008, '52-Week Low':
0.4378, 'Average True Range': 0.86, 'EPS growth next year': 0.1194, 'Sales
growth past 5 years': 0.048, 'Company': 'Agilent Technologies Inc.', 'Gap':
0, 'Relative Volume': 0.79, 'Volatility (Month)': 0.0168, 'Market Cap': 173
```

Question 5

Expected answer:

```
['Agilent Technologies Inc.',
 'Alcoa, Inc.',
 'WCM/BNY Mellon Focused Growth ADR ETF',
 'iShares MSCI AC Asia Information Tech',
 'Altisource Asset Management Corporation',
 'Atlantic American Corp.',
 "Aaron's, Inc.",
 'Applied Optoelectronics, Inc.',
 'AAON Inc.',
 'Advance Auto Parts Inc.']
```

In [12]:
```
# from records, extract the first 10 company names and store in 'companies'

# your code here
companies = []
for i,x in enumerate(record):
    if i == 10:
        break
    else:
        companies.append(x['Company'])
print(companies)
```

```
['Agilent Technologies Inc.', 'Alcoa, Inc.', 'WCM/BNY Mellon Focused Growth A
DR ETF', 'iShares MSCI AC Asia Information Tech', 'Altisource Asset Managemen
t Corporation', 'Atlantic American Corp.', "Aaron's, Inc.", 'Applied Optoelec
tronics, Inc.', 'AAON Inc.', 'Advance Auto Parts Inc.']
```

Question 6

Expected answer:

```
['Agilent Technologies Inc.',
 'Alcoa, Inc.',
 "Aaron's, Inc.",
 'Applied Optoelectronics, Inc.',
 'AAON Inc.',
 'Advance Auto Parts Inc.']
```

In [13]:
```
# from the top 10 companies, show all the companies with the word 'Inc.'

# your code here
incCompany=(list(filter(lambda x:x.find('Inc')>=0,companies)))
print(incCompany)
```

```
['Agilent Technologies Inc.', 'Alcoa, Inc.', "Aaron's, Inc.", 'Applied Optoel
ectronics, Inc.', 'AAON Inc.', 'Advance Auto Parts Inc.']
```

Question 7

Expected answer:

```
41.71060205580027
```

In [36]:
```
# get the average 'P/E' for all data

# your code here
sum = 0
count=0
for i in record:
    if i.get("P/E") is not None:
        count+=1
        sum+=i['P/E']
print(sum/count)
```

```
41.71060205580027
```