

Program Code: J620-002-4:2020

**Program Name: FRONT-END SOFTWARE DEVELOPMENT** 

**Title: List, Tuple and Dictionary** 

Name: Ooi Caaron

IC Number: 990701-07-5837

Date: 24/6/23

Introduction: Using List, Tuple and Dictionary to return the output

Conclusion: Still need to do a lot practice

## **EXERCISE 4**

## **List, Tuple and Dictionary**

```
In [ ]:
```

Note : Please start your jupyter notebook using the anaconda prompt with this command to Data Rate Exceeded Problem  ${\sf Note}$ 

At the anaconda prompt, type : jupyter notebook --NotebookApp.iopub\_data\_rate\_limit=1.0e1

### **Question 1**

Expected answer:

match\_ends

3

2

1

```
In [10]:
```

```
# 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
    count = 0
    for i in words:
        if len(i) >= 2:
             if i[0] == i[-1]:
                 count +=1
    return count
print('match ends')
print(match_ends(text1))
print(match_ends(text2))
print(match_ends(text3))
```

# match\_ends

2

1

## **Question 2**

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

```
In [16]:
```

```
# 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
    new = []
    s = sorted(words)
    for i in s:
         if i[0] == "x":
             new.append(i)
    for i in s:
         if i[0] != "x":
             new.append(i)
    return new
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**

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

#### In [10]:

```
# 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 n : n[-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 [33]:

```
# read the stocks.json file and store into 'records'
import json

%pwd

path ='stocks.json'

stock_data = []
with open(path) as f:
    for line in f:
        stock_data.append(json.loads(line))

records = stock_data
print(records)
```

[{'\_id': {'\$oid': '52853800bb1177ca391c17ff'}, 'Ticker': 'A', 'Profit Margin': 0.137, 'Institutional Ownership': 0.847, 'EPS growth past 5 year s': 0.158, 'Total Debt/Equity': 0.56, 'Current Ratio': 3, 'Return on Ass ets': 0.089, 'Sector': 'Healthcare', 'P/S': 2.54, 'Change from Open': -0.0148, 'Performance (YTD)': 0.2605, 'Performance (Week)': 0.0031, 'Quic k Ratio': 2.3, 'Insider Transactions': -0.1352, 'P/B': 3.63, 'EPS growth quarter over quarter': -0.29, 'Payout Ratio': 0.162, 'Performance (Quart er)': 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 High': -0.0544, 'P/Cash': 7.45, 'Change': -0.01 48, 'Analyst Recom': 1.6, 'Volatility (Week)': 0.0177, 'Country': 'USA', 'Return on Equity': 0.182, '50-Day Low': 0.0728, 'Price': 50.44, '50-Day High': -0.0544, 'Return on Investment': 0.163, 'Shares Float': 330.21, 'Dividend Yield': 0.0094, 'EPS growth next 5 years': 0.0843, 'Industry': 'Medical Laboratories & Research', 'Beta': 1.5, 'Sales growth quarter ov er quarter': -0.041, 'Operating Margin': 0.187, 'EPS (ttm)': 2.68, 'PE G': 2.27, 'Float Short': 0.008, '52-Week Low': 0.4378, 'Average True Ran ge': 0.86, 'EPS growth next year': 0.1194, 'Sales growth past 5 years': 0.048, 'Company': 'Agilent Technologies Inc.', 'Gap': 0, 'Relative Volum

## **Question 5**

```
['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 [11]:

```
# from records, extract the first 10 company names and store in 'companies'
# print(records[:10])
# your code here
data = records[:10]
companies = []
for i in data:
    companies.append(i["Company"])
print(companies)
```

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

### **Question 6**

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

```
In [23]:
```

```
# from the top 10 companies, show all the companies with the word 'Inc.'

# your code here
companies = []
inc = []
str=""
for i in records:
    companies.append(i["Company"].split())

for i in companies:
    for j in i:
        if j == "Inc.":
             inc.append(str.join(i))
```

['AgilentTechnologiesInc.', 'Alcoa,Inc.', "Aaron's,Inc.", 'AppliedOptoel ectronics, Inc.', 'AAONInc.', 'AdvanceAutoPartsInc.', 'AppleInc.', 'Ameri canAssetsTrust,Inc.', 'AtlasAirWorldwideHoldingsInc.', 'AbaxisInc.', 'Ab bVieInc.', 'CambiumLearningGroup,Inc.', 'AsburyAutomotiveGroup,Inc.', 'ARCAbiopharma,Inc.', 'ABMIndustriesInc.', 'AbiomedInc.', 'ArborRealtyTrus tInc.', 'AutobytelInc.', 'ACADIAPharmaceuticals, Inc.', 'ArcticCatInc.', 'AmericanCampusCommunitiesInc.', 'AccelrysInc.', 'AcornEnergy,Inc.', 'Al lianceBernsteinIncomeFund,Inc.', 'AcadiaHealthcareCompany,Inc.', 'Achill ionPharmaceuticals,Inc.', 'ArchCoalInc.', 'ACIWorldwide,Inc.', 'AxcelisT echnologiesInc.', 'AcordaTherapeutics, Inc.', 'ActivePowerInc.', 'AcelRxP harmaceuticals, Inc.', 'AcastiPharmaInc.', 'TheActiveNetwork, Inc.', 'Acur aPharmaceuticalsInc.', 'AdobeSystemsInc.', 'AdeptTechnologyInc.', 'ADA-E S,Inc.', 'AmericanDGEnergy,Inc.', 'AnalogDevicesInc.', 'AdCareHealthSyst emsInc.', 'Audience,Inc.', 'AutomaticDataProcessing,Inc.', 'Autodesk,In c.', 'ADTRANInc.', 'AdventSoftware,Inc.', 'Advaxis,Inc.', 'AdamsResource s&EnergyInc.', 'AegerionPharmaceuticals,Inc.', 'AdvancedEnergyIndustrie s,Inc.', 'AmericanEagleOutfitters,Inc.', 'AmericanElectricPowerCo.,In c.', 'AEPIndustriesInc.', 'AppliedEnergetics,Inc.', 'AeriePharmaceutical s,Inc.', 'AdvancedEnvironmentalRecyclingTechnologies,Inc.', 'AetnaInc.',

### **Question 7**

Expected answer:

41.71060205580027

### In [48]:

```
# get the average 'P/E' for all data

# your code here
pe = []
for i in records:
    if 'P/E' in i:
        pe.append(i['P/E'])
total = sum(pe)
print(total/len(pe))
```

#### 41.71060205580027

### In [ ]: