Collections

Lists, tuples, sets, dict

List

A list is a compound data type that contains a collection of **sequential** related data items

Key properties

* collection of related objects
* ordered or sequential collection
* mutable. can be modified

Examples

```
list_of_names = [“John”,”Jack”,”Jill”,”Joan”]
list_of_tickers = [“AAPL”,”IONS”,”GE”,”DB”]
list_of_natural_numbers = [1,2,3,4,5,6,7]
```

List operations

```
long_list = [1,['a',['b','c']],43,"Too many cooks spoil the broth"]
long_list.append('Many hands make light work') #adds an item to the back of the list
long_list[3] #Gets the 4th item in the list
long_list[1][1][0] #Accessing nested items
long_list.extend(['e','f']) #appends contents of a list
long_list.remove(1) #Removes the item with the VALUE 1
long_list.pop() #Removes and returns the last item
long_list.pop(1) #Removes and returns the ith item
len(long_list) #Returns the length of the list
```

List are mutable

Contents of a list can be changed

Examples

```
x = [1,2,3,4]
x[0]=8  --> [8,2,3,4]
```

Tuples

Tuples are **immutable** sequential data collections

price = (“20150904”,545.23)
price[0] —> “20140904”
price[1] —-> 545.23
price[1]=26.3 —-> TypeError
price[2] —-> IndexError

Tuples are just like lists except they are not mutable (cannot be changed)!

list of tuples

```
prices = [(‘AAPL',96.43),('IONS',39.28),('GS',159.53)]
```

Question: Print all tickers and prices

Question: What is the price of IONS?

Question: Which is the most expensive stock?

Iteration

for index in range

the **for index in range** construct makes iteration possible

range: a **sequence** of integers from 0 to length of prices

len: the number of items in prices

```
for index in range(len(prices)):
    print(prices[index][0],prices[index][1])
```

for: tells python to expect a repetition block

index: a variable name that holds each value of the sequence in turn. One iteration - one value!

for item in sequence

generalizing:

the **for something in some_sequence** construct makes iteration possible

stock_price: a variable that will map to each element in the list sequentially

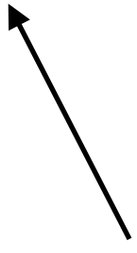
```
prices = [('AAPL',96.43),('IONS',39.28),('GS',159.53)]
for stock_price in prices:
    print(stock_price[0],stock_price[1])
```

for: an iteration statement

break and else

break: iteration ends when a break is encountered

```
prices = [('AAPL',96.43),('IONS',39.28),('GS',159.53)]
ticker = input('Please enter a ticker: ')
for item in prices:
    if item[0] == ticker:
        print(ticker,item[1])
        break
else:
    print("Sorry",ticker,"was not found in my database")
```



else: the program will do this only if the while does not encounter a 'break'

Practice

Write a function search_list that searches a list of tuple pairs and returns the value associated with the first element of the pair

```
prices = [('AAPL',96.43),('IONS',39.28),('GS',159.53)]
x=search_list(prices,'AAPL')
#The value of x should be 96.43
x=search_list(prices,'GOOG')
#The value of x should be None

inventory = [('widgets',100),('spam',30),('eggs',200)]
y=search_list(inventory,'spam')
#The value of y should be 30
y=search_list(prices,'hay')
#The value of y should be None
```

dictionaries

key - value pairs
retrieval through **lookups**
just like real dictionaries!

```
mktcaps = {‘AAPL’:538.7,’GOOG’:68.7,'IONS':4.6}
mktcaps['AAPL'] #key-based retrieval
print(mktcaps[‘AAPL’])
mktcaps['GE'] #error (no "GE")
'GE' in  mktcaps
mktcaps.keys() #returns a list of keys
sorted(mktcaps.keys()) #returns a sorted list of keys
```

Sets

Sets are **unordered** collections of **unique** objects
-tickers, markets, products

```
tickers={“AAPL”,"GE","NFLX","IONS"}
regions={“North East”,”South","West coast”,"Mid-West"}
"AAPL" in tickers #membership test
"IBM" not in tickers #non-membership test
pharma_tickers={"IONS","IMCL"}
tickers.isdisjoint(pharma_tickers) #empty intersection
pharma_tickers <= tickers #subset test
pharma_tickers <tickers  #proper-subset test
tickers  > pharma_tickers #superset
tickers  & pharma_tickers #intersection
tickers  | pharma_tickers #union
tickers  - pharma_tickers #set difference
```