ClassPractice-Dictionaries

October 7, 2024

1 In-Class Coding : Dictionaries

The goals of this lab are to help you understand:

- How to use Python Dictionaries
- Basic Dictionary methods
- Dealing with Key errors
- How to use lists of Dictionaries
- How to encode / decode python dictionaries to json.

1.1 Dictionaries are Key-Value Pairs.

The **key** is unique for each Python dictionary object and is always of type `str`. The **value** stored under the key can be any Python type.

This example creates a `stock` variable with two keys `symbol` and `name`. We access the dictionary key with `['keyname']`.

[ ]:
```
stock = {} # empty dictionary
stock['symbol'] = 'AAPL'
stock['name'] = 'Apple Computer'
print(stock)
print(stock['symbol'])
print(stock['name'])
```

While Python lists are best suited for storing multiple values of the same type (like grades), Python dictionaries are best suited for storing hybrid values, or values with multiple attributes.

In the example above we created an empty dictionary `{}` then assigned keys `symbol` and `name` as part of individual assignment statements.

We can also build the dictionary in a single statement, like this:

[ ]:
```
stock = { 'name' : 'Apple Computer', 'symbol' : 'AAPL', 'value' : 125.6 }
print(f"Dictionary: {stock}")
print(f"{stock['name']} ({stock['symbol']}) has a value of ${stock['value']:.
↪2f}")
```

1.2 Dictionaries are mutable

This means we can change their value. We can add and remove keys and update the value of keys. This makes dictionaries quite useful for storing data.

[ ]:
```
# let's add 2 new keys
print("Before changes", stock)
stock['low'] = 119.85
stock['high'] = 127.0

# and update the value key
stock['value'] = 126.25
print("After change", stock)
```

1.2.1 1.1 You Code

Create an empty python dictionary variable called `car` with the following keys `make`, `model` and `price`. Set appropriate values and print out the dictionary.

[ ]:
```
# TODO: Write code here
```

1.3 What Happens when the key is not there?

Let's go back to our stock example. What happens when we try to read a key not present in the dictionary?

The answer is that Python will report a `KeyError`

[ ]:
```
print("Keys", stock.keys())
print( stock['change'] )
```

No worries. We know how to handle run-time errors in Python... use `try except` !!!

[ ]:
```
try:
    print( stock['change'] )
except KeyError:
    print("The key 'change' does not exist!")
```

1.4 Using the `get()` method to avoid KeyError

You can avoid `KeyError` using the `get()` dictionary method. This method will return a default value when the key does not exist.

The first argument to `get()` is the key to get, the second argument is the value to return when the key does not exist.

[ ]:
```
print(f"KEY: name, VALUE",stock.get('name','no key'))
print(f"KEY: change, VALUE",stock.get('change', 'no key'))
```

1.4.1 1.2 You Code

Write a program to ask the user to input a key for the `stock` variable. If the key exists, print the value, otherwise print 'Key does not exist'. You can use the `get()` method or `try..except`

Sample Run #1

```
Enter key: symbol
KEY: symbol, VALUE: AAPL
```

Sample Run #2

```
Enter key: mike
KEY: mike, VALUE: Key Does Not exist
```

[ ]:
```
stock = { 'name' : 'Apple Computer', 'symbol' : 'AAPL', 'value' : 125.6 }
# TODO: write code here
```

1.5 Enumerating keys and values

You can enumerate keys and values easily, using the `keys()` and `values()` methods:

[ ]:
```
print("KEYS")
for k in stock.keys():
    print(k)

print("VALUES")
for v in stock.values():
    print(v)
```

1.5.1 1.3 You Code: Debug

The following program should loop over the keys in a dictionary and print the keys and values. Debug this code to get it working.

Expected output (when code it corrected):

```
KEY: make VALUE: tesla
KEY: model VALUE: S
KEY: price VALUE: 69420
KEY: owner VALUE: bob
KEY: location VALUE: Syracuse, NY
KEY: moving VALUE: False
```

[ ]:
```
car = {'make': 'tesla', 'model': 'S', 'price': 69420, 'owner': 'bob',␣
 ↪'location' : 'Syracuse, NY', 'moving' : False }
# TODO: Debug this code
for key in car:
    value = car['key']
    print(f"something")
```

1.6 List of Dictionary

The List of Dictionary object in Python allows us to create useful in-memory data structures. It's one of the features of Python that sets it apart from other programming languages.

Let's use it to build a portfolio (list of 4 stocks).

[ ]:
```
portfolio = [
    { 'symbol' : 'AAPL', 'name' : 'Apple Computer Corp.', 'value': 136.66 },
    { 'symbol' : 'AMZN', 'name' : 'Amazon.com, Inc.', 'value': 845.24 },
    { 'symbol' : 'MSFT', 'name' : 'Microsoft Corporation', 'value': 64.62 },
    { 'symbol' : 'TSLA', 'name' : 'Tesla, Inc.', 'value': 257.00 }
]

print("first stock", portfolio[0])
print("name of first stock", portfolio[0]['name'])
print("last stock", portfolio[-1])
print("value of 2nd stock",  portfolio[1]['value'])
```

[ ]:
```
print("Here's a loop:")
for stock in portfolio:
    print(f"  {stock['name']} ${stock['value']}")
```

1.7 Using `json` to read in lists of dictionary

JSON (JavaScript Object Notation) files can be read in using Python's `json` module, and de-serialized into a list of dictionary. This is very powerful feature of the Python programming language and would require a lot more code to accomplish in other programming languages. It is one of the reasons Python is so attractive for the web and data manipulation.

Run this code to download the json file

[ ]:
```
!curl https://raw.githubusercontent.com/mafudge/datasets/master/ist256/
↪09-Dictionaries/stocks.json -o stocks.json
```

This code will read in the json file `stocks.json` and create a list of dictionary, `stocks`

[ ]:
```
import json
with open('stocks.json','r') as f:
    stocks = json.load(f)

stocks
```

1.8 What is so special about a list of dictionary?

You might be thinking "so what" or "big deal". Actually those 3 lines of Python code are a big deal. We created a variable `stocks` from a json text file. All the heavy lifting of tokenizing and parsing have been done for us!

We can now use Python code to access the data. For example, to get `IBM`'s stock price we can loop over the list until we find the `symbol IBM`:

[ ]:
```
find = 'IBM'
for stock in stocks:
    if stock['symbol'] == find:
        print(f"symbol: {find} price: ${stock['price']:.2f}")
```

1.8.1 1.4 You code

Build upon the code in the previous cell. Write a program to input a symbol and print out the price of that symbol. If the symbol is not in the list of stocks, print `Symbol not found.`

Example Run #1

```
Enter a stock symbol: MSFT
symbol: MSFT price: $212.55
```

Example Run #2

```
Enter a stock symbol: ABCD
Symbol not found.
```

[ ]:
```
import json
with open('stocks.json','r') as f:
    stocks = json.load(f)

# TODO Write code here:
```

1.9 Putting It All Together

Write a program to build out your personal stock portfolio.

```
1. Start with an empty list, called `mystocks`
2. call `loadStocks()` to get a list of stocks from the file.
2. loop
3.     input a stock symbol, or type 'QUIT' to print your portfolio
4.     if symbol equals 'QUIT' exit loop
5.     call `findStock()` to locate the input symbol in the list of stocks.
6.     if you found the stock (it's not an empty dictionary) then add it to the `mystocks` lis
9. time to print the portfolio: for each stock in the `mystocks`
10.    print stock symbol and stock value, like this "AAPL $136.66"
```

We will write few `loadStocks()` and `findStocks()`functions for you to make this a bit more manageable.

Sample Run:

```
Enter stock symbol to add to your portfolio or quit: IBM
Enter stock symbol to add to your portfolio or quit: FB
Enter stock symbol to add to your portfolio or quit: quit
Your Portfolio
```

IBM $128.39

FB $251.11

1.9.1 1.5 You Code

[ ]:
```
def loadStocks():
    import json
    with open('stocks.json','r') as f:
        stocks = json.load(f)
        return stocks

def findStock(symbolToFind):
    found = {}
    for stock in stocks:
        if stock['symbol'] == symbolToFind:
            return stock

# TODO: Wwite code here
```

2 Metacognition

2.0.1 Rate your comfort level with this week's material so far.

1 ==> I don't understand this at all yet and need extra help. If you choose this please try to articulate that which you do not understand to the best of your ability in the questions and comments section below.
2 ==> I can do this with help or guidance from other people or resources. If you choose this level, please indicate HOW this person helped you in the questions and comments section below.
3 ==> I can do this on my own without any help.
4 ==> I can do this on my own and can explain/teach how to do it to others.

--== Double-Click Here then Enter a Number 1 through 4 Below This Line ==--