Rotman

STORING DATA

February 8, 2022

Prepared by Niti
TDMDAL & FinHUB

Rotman School of Management
UNIVERSITY OF TORONTO

List

Lists Definition

- Lists are another sequence data type used for data collection.
- Lists contain elements that can be of any datatype, including another list.
- Lists are ordered sequences and can be indexed.
- Lists are mutable data type and can be changed once created.

Anatomy of a Python List

LIST

- Mutable
- Ordered
- Sequence of items

```
tickers = ['aapl', 'jpm' ,
           'spy']
```

Each element separated by comma.

All elements contained inside square brackets.

Rotman

Modifying Lists

- The index position of an element in a list can be used to modify the value of that element.
- Various methods can also be used to modify elements of a list.
- Adding elements, deleting element or updating existing elements in a list are example of list modification.

Lists Operations

- Arithmetic operations on list behave similar to such operations on string.

- Membership operator can be used to check the existence of an element in a list.

```
In [1]:  prices = [126, 145, 387]
         126 in prices

Out[1]: True

In [2]:  150 in prices

Out[2]: False
```


Rotman

Looping through Lists

- Each element in a list can be manipulated succinctly by writing a loop.

```
prices = [126,149,387]
for i in prices:
    print(i)
    print(i*.3)
    print('='*20)
```

```
126
37.8
====================
149
44.699999999999996
====================
387
116.1
====================
```


Rotman

Lists are Modified Inplace

- Unless stated otherwise, majority of the manipulations of list using methods change the original list.

- This is because, unlike strings, lists are mutable.

```
In [1]:  prices = [126, 149, 387]
         prices.reverse()

In [2]:  prices

Out[2]:  [387, 149, 126]
```

Notice that assignment operator has not been used.

Yet, the list *prices* has changed "inplace".

Rotman

Nested Lists

- Nested lists are lists that contain list(s) inside of it.

```
In [1]:  prices = [ [126, 149, 387], ['aapl', 'jpm', 'spy'] ]
         prices[0]

Out[1]: [126, 149, 387]

In [2]:  prices[1][0]

Out[2]: 'aapl'
```

Example of indexing a list inside of another list.

Rotman

Tuples

Tuples Definition

- Similar to lists, tuples are also ordered sequences and thus, can be indexed.
- Unlike lists, tuples are immutable, meaning you cannot add, delete or change elements of a tuple.
- Tuples have few methods available to them.

Anatomy of a Tuple

- Tuples follow the same anatomy as lists, except for the square brackets used in lists, are replaced by parenthesis in tuples.

Anatomy of a Tuple

TUPLE

- Immutable
- Ordered
- Sequence of items

```
net2021 = (2247.12, 391.10,
           37.418)
```

All elements contained inside square brackets.

Rotman

Dictionary

What is a Dictionary?

- A Dictionary is a collection of key-value pairs.
- A Dictionary is unordered and mutable.
- Keys in a dictionary are unique and maps to its corresponding value.
- Keys can be string, number or even tuple but not list.
- Values can be any data type.

Anatomy of a Dictionary

DICTIONARY

- Mutable
- Unordered
- Key-Value Pairs

keys values

```
comp_info = {'aapl': 126,
             'jpm': 149,
             'spy': 387}
```

Keys and their values are separated by colon

All key-value pairs are contained inside curly brackets.

Rotman

Indexing a Dictionary

- Items in dictionaries are unordered and thus, do not have index position.
- To access items of a dictionary, we need to use its keys instead.
- Keys are enclosed inside the index operator to obtain the corresponding value.

```
In [1]:  comp_info = {'aapl': 126, 'jpm':149, 'spy':387}
         comp_info['jpm']

Out[1]:  149
```

key

value of that key

Rotman

Modifying a Dictionary

- Key-value pairs can be added and deleted from a dictionary.

```
In [1]:  comp_info = {'aapl': 126, 'jpm':149, 'spy':387}
         comp_info['jpm']

Out[1]:  149
```

```
In [2]:  # delete a key-value pair from a dictionary
         del comp_info['jpm']
         comp_info

Out[2]:  {'aapl': 126, 'spy': 387}
```

- Indexing operators can be used to change a dictionary.

```
In [3]:  # add key a key-value pair in a dictionary
         comp_info['msft'] = 215
         comp_info

Out[3]:  {'aapl': 126, 'spy': 387, 'msft': 215}
```

- Methods are also available to add, delete or change key-value pairs in a dictionary.

```
In [4]:  # update value of key msft in dictionary
         comp_info.update({'msft':216})
         comp_info

Out[4]:  {'aapl': 126, 'spy': 387, 'msft': 216}
```


Rotman

List as values of a Dictionary

- Dictionary values can also be a list.

```
In [1]:  stockprices = {'name':['aapl', 'jpm', 'spy'], 'prices':[126, 149, 387]}
         stockprices['name']

Out[1]:  ['aapl', 'jpm', 'spy']
```

- Both dictionary and list indexing methods can be used to extract items from such dictionaries.

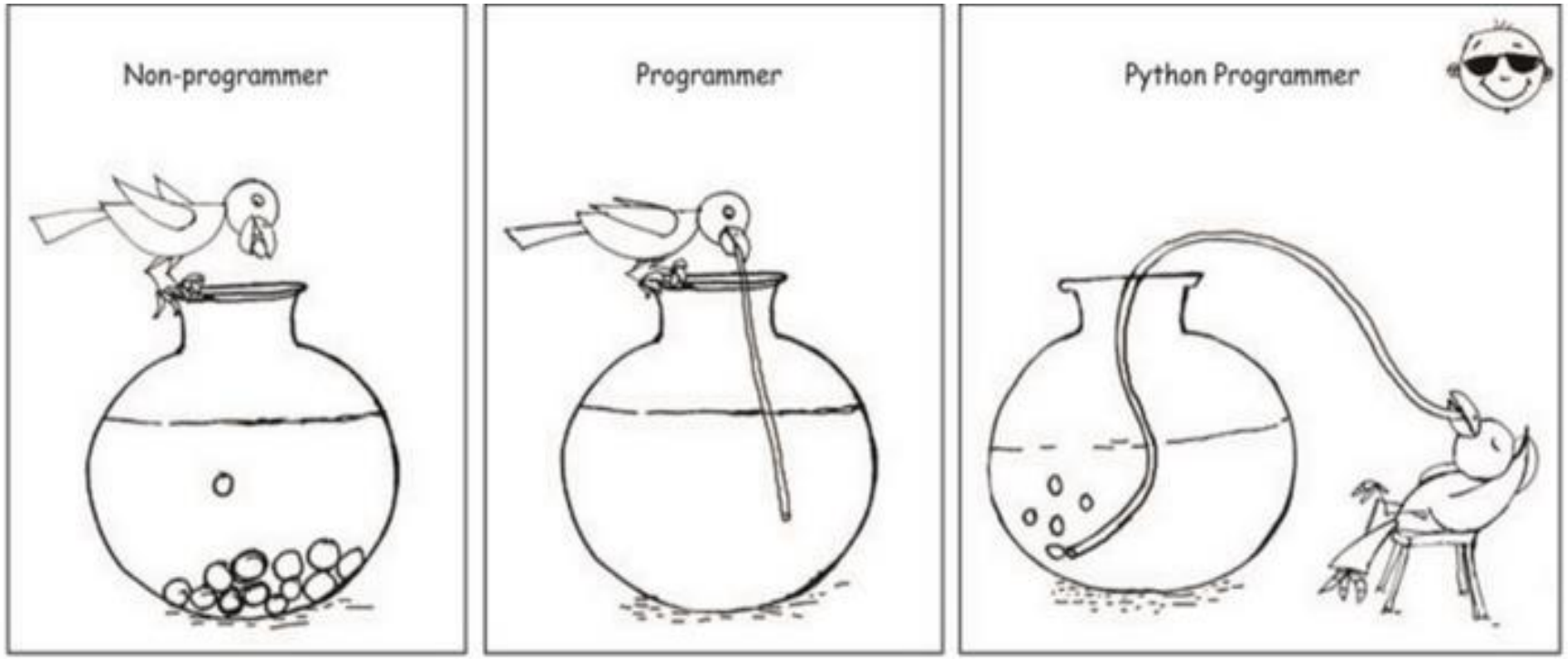
```
In [2]:  stockprices['name'][0]

Out[2]:  'aapl'

In [3]:  stockprices['prices'][0]

Out[3]:  126
```

dictionary indexing

list indexing

Rotman

Questions?



Who wants to become a Python Programmer?

Thank you