Intermediate Python

Research Data Services

Lesson Plan

Data Structures

- Lists
- Dictionaries
- Tuple (Brief)
- Set (Brief)

For Loops

- Lists
- Dictionaries
- Files

Import Libraries

- Pip Install

Open Files

Data Structures

There are 4 commonly used built-in data structures in python [1]:

- lists
- dictionaries
- tuples
- sets

Data structures can store any of the basic data types

[1] https://github.com/jakevdp/WhirlwindTourOfPython/blob/master/06-Built-in-Data-Structures.ipynb

Lists

Lists are a variable that can store multiple items

Syntax: `myList = [ ]`

What separates lists from other data structures is:

- Order Matters
- Can have duplicates
- Can be changed

https://docs.python.org/3/tutorial/datastructures.html

First List

```
myList = ["apple", "orange", "apple", "banana"]
```

lists are denoted by [] with commas

```
print(myList)
```

```
['apple', 'orange', 'apple', 'banana']
```

Indexing

Extracting specific data.

Index starts with 0

We can do 3 main things with indexing:

- Regular Indexing
- Negative Indexing
- Ranges

Indexing

First Item

```
print(myList[0])
```

```
apple
```

Last Item

```
print(myList[-1])
```

```
banana
```

Ranges

```
print(myList[1:3])
```

```
['orange', 'apple']
```

```
print(myList[1:-1])
```

```
['orange', 'apple']
```

Common List Functions

- **len(), outputs size of list**
- **changing values**
- **insert(), inputs value at given index**
- **append(), adds to end of list**
- **extend(), adds list to end of list**
- **remove(), removes given item**
- **pop(), removes index**
- del, removes index
- clear(), clear list
- copy(), copy lists
- count(), outputs number of certain value
- index(), returns index of given value
- sorts(), sorts list
- reverse(), reverse list

https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range

Exercise One

Create a list of with:

- at least 5 vegetables
- sort it in alphabetical order
- print out the third item
- then clear the list

Dictionaries

Dictionaries use key and value pairs

Syntax: `myDictionary = {key: value}`

What separates dictionaries from the other storage types is:

- Indexing is done with the keys
- It is ordered
- Can't have duplicates
- Changeable

First Dictionary

```
myDictionary = {"name": "John", "age":
25, "height":5.123, "adult":False}

print(myDictionary)

print(myDictionary[0])

print(myDictionary["age"])
```

```
{'name': 'John', 'age': 25, 'height':
5.123, 'adult': True}

KeyError: 0

25
```

https://docs.python.org/3/tutorial/datastructures.html#dictionaries

Indexing

As we saw above, dictionaries are not indexed with index values, but with their keys

There are 3 functions that are useful for looking at the data stored in dictionaries

- keys()
- values()
- items()

Indexing Functions

```
keys = myDictionary.keys()

print(keys)

values = myDictionary.values()

print(values)

items = myDictionary.items()

print(items)
```

```
dict_keys(['name', 'age', 'height',
'adult'])


dict_values(['Avery', 19, 5.583, True])


dict_items([('name', 'Avery'), ('age',
19), ('height', 5.583), ('adult', True)])
```

Common Dictionary Functions

- copy(), copy dictionary
- changing values
- get(), grabs the value of the given pair
- update(), changes the values of the key value pair or adds the value
- popitem(), removes last item inserted in
- pop(), removes the item with the given key
- del, removes index
- dict(), can be used to copy
- clear(), clear dictionary

https://docs.python.org/3/library/stdtypes.html

Exercise 2

- Create a dictionary 5 food types and their foods like soda and sprite.
- Print one of your values using key indexing.
- Clear the dictionary, do not delete it

Tuples

A storage type that stores preset values

Syntax: `myTuple = ()`

Properties include:

* Cannot be changed

https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences

Tuple Examples

```
my_tuple = (1, 3, 5)

print(my_tuple)

my_tuple[0] = 10
```

```
(1, 3, 5)

TypeError: 'tuple' object does
not support item assignment
```

Sets

One use-case is in removing duplicates of data

Syntax: `sets = {}`

Properties:

- No duplicates allowed
- Unordered
- No index

For more examples of sets, see:

https://github.com/jakevdp/WhirlwindTourOfPython/blob/master/06-Built-in-Data-Structures.ipynb

Syntax Example

```
sets = {"apple", "banana",
"banana", "orange"}

print(sets)
```

```
{'apple', 'banana', 'orange'}
```

Multi-dimensional Storage Types

While it sounds complicated, it is just put storage types in other storage types:

- Dictionaries in Lists
- Lists in Dictionaries
- Etc.

https://nbviewer.org/github/jakevdp/WhirlwindTourOfPython/blob/master/06-Built-in-Data-Structures.ipynb

For Loops

With python, looping through these data sets will be a lot easier.

for in range() is just looping through a list

```
print(list(range(10)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

https://nbviewer.org/github/jakevdp/WhirlwindTourOfPython/blob/master/07-Control-Flow-Statements.ipynb

For Loops

Looping is practically all the same, just slightly different

Types of For Loops:

- Lists
- Dictionaries
- Multi-Dimensional

For Loops: Lists

For loops with list can be achieved 2 ways:

- values
- indexes

Lists: Value For Loop

Just like for in range():

Syntax: `for {variableName} in {listName}:`

Exercise 3

Looping through the List and print out each value

```
myList = ["Lists", "Dictionaries", "Sets", "Tuples"]
```

Lists: Value For Loop

```
myList = ["Lists", "Dictionaries", "Sets",
"Tuples"]

for storageType in myList:

   print(storageType)
```

```
Lists

Dictionaries

Sets

Tuples
```

Lists: Index For Loop

We will loop through using the index values

Syntax: `for {variableName} in range(len({listName})):`

Or

```
for idx, variableName in enumerate(listName):
```

Lists: Index For Loop

```
myList = ["Lists", "Dictionaries", "Sets",
"Tuples"]

for index in range(len(myList)):

   print(index, myList[index])
```

```
0 Lists

1 Dictionaries

2 Sets

3 Tuples
```

```
for idx, storageType in enumerate(myList):

   print(idx, storageType)
```

For Loops List of Lists

```
students = [
   ["Daniel", "Sophomore",[90,20,100]],
   ["Tristan","Junior", [100,65,87]]
]
for student in students:
   print(student)
```

```
['Daniel', 'Sophmore', [90, 20, 30]]
['Tristan', 'Junior', [100, 65, 38]]
```

https://docs.python.org/3/tutorial/datastructures.html#nested-list-comprehensions

Exercise 4

loop through a lists of list and print only the scores

```
students = [
   ["Daniel", "Sophomore",[90,20,100]],
   ["Tristan","Junior", [100,65,87]]
]
```

For Loops: Dictionaries

You can loop through multiple things based off what you want to do

* keys

* pairs

Dictionaries: Keys For Loop

Syntax: `for {variableName} in {dictionary}:`

keys Loop

```
states = {

   "AL": "Alabama",

   "AK": "Alaska",

   "AZ": "Arizona",

   "AR": "Arkansas"

}

for abbreviations in states:

   print(abbreviations,
states[abbreviations])
```

AL Alabama

AK Alaska

AZ Arizona

AR Arkansas

Dictionaries: Pair For Loop

Syntax: `for key,value in dictionary.items():`

Pairs Loop

```
states = {
   "AL": "Alabama",
   "AK": "Alaska",
   "AZ": "Arizona",
   "AR": "Arkansas"
}
for abbreviations,state in states.items():
   print(abbreviations, state)
```

```
AL Alabama
AK Alaska
AZ Arizona
AR Arkansas
```

Exercise 5

Loop through the dictionary in a list

Print out the key and value on the same line: `name Daniel`

```
students = [
    {
        "name": "Daniel",
        "year": "Sophmore"
    },
    {
        "name": "Tristan",
        "year": "Junior"
    }
]
```

Importing Libraries

What makes python super powerful is that we can import vast amounts for libraries for almost any use case

We will cover:

- Pip Installing
- Importing
- Managing Files

https://github.com/ualibweb/UALIB_Workshops/tree/master/06_Conda_fall_2022

Installing Libraries

We will be installing Numpy

1. Open up the terminal
2. Type ```pip install numpy```

Importing

Now we just have to tell python that we are going to use the library

```
import numpy

print(numpy.random.randint(100))
```

69

```
from numpy import random

print(random.randint(100))
```

2

```
import numpy as np

print(np.random.randint(100))
```

53

Managing Files

Now that we can download libraries and use them, let's quickly cover how to keep track of them

- Find Packages
- List Packages
- Remove Packages

Find Packages

https://pypi.org/

List Packages

pip list

Remove Packages

pip uninstall {packageName}

Files

Files allows us to store data for later uses or to import data

- Reading Files
- Writing Files
- Deleting Files
- Looping

https://docs.python.org/3/tutorial/inputoutput.html

Reading Files

Syntax: `open({fileName}, "r")`

Functions:

- read()
- readline()
- close()

Writing Files

- Writing to a file
- Appending to a file

Looping through files

This allows us to go through each individual line

```
for line in open("02_text_file.txt"):

   print(line)
```

```
with open("02_text_file.txt", "r") as
inFile:

   for line in inFile:

       print(line)
```

Exercise 6

Create a List with at least 5 elements

Loop through the list

Write elements to a file, each element on its own line