ENGL 516X:
Methods of Formal Linguistic Analysis

Semester: Spring '18

Instructor: Sowmya Vajjala

Iowa State University, USA

22 Feb 2018

Class outline

- Files: Wrap up
- Dictionaries and Tuples in Python
- Tuples continued
- How to work with so many different data structures
- Other datastructures (general overview)
- Project ideas for final exam

Files: Last Class' Question

- Reviewing ListFilesInDir.py

Files: Last Class' Question

- Reviewing ListFilesInDir.py
- Changing something there: using os.path.join() method.
- Useful ref:
  `https://docs.python.org/3.1/library/os.path.html`

Files: Last Class' Question

- Reviewing ListFilesInDir.py
- Changing something there: using os.path.join() method.
- Useful ref:
  `https://docs.python.org/3.1/library/os.path.html`
- Question: Counting number of lines in each file in a directory - solution: CountLinesOfFilesInDirectory.py on Canvas

Files: Last Class' Question

- Reviewing ListFilesInDir.py
- Changing something there: using os.path.join() method.
- Useful ref:
  `https://docs.python.org/3.1/library/os.path.html`
- Question: Counting number of lines in each file in a directory - solution: CountLinesOfFilesInDirectory.py on Canvas
- Did anyone try to use a pdf/doc file in open(filename).read()?

On the discussion forum

- I am not discussing some of the solutions as there are multiple answers posted on the forum
- Thanks to all the active participants there!
- Please take a look once in a while, and try to understand other people's programs. Ask questions if you want.
- Reading and understanding other people's code is going to be very important once you stop doing classwork!

Dictionaries in Python

1. Dictionaries are a way of storing data as pairs (called: "key-value" pair).

Dictionaries in Python

1. Dictionaries are a way of storing data as pairs (called: "key-value" pair).
2. Think of a telephone directory - we access it by names (keys) to get numbers (values).
3. An Example dictionary structure: eg = {'Sowmya': 'India', 'Taichi': 'Japan', 'Nazlinur': 'Turkey', 'Brody':'USA'}

Dictionaries in Python

1. Dictionaries are a way of storing data as pairs (called: "key-value" pair).
2. Think of a telephone directory - we access it by names (keys) to get numbers (values).
3. An Example dictionary structure: eg = {'Sowmya': 'India', 'Taichi': 'Japan', 'Nazlinur': 'Turkey', 'Brody':'USA'}
   In this, eg['Sowmya'] returns me 'India'.
4. A dictionary can have a dictionary embedded within itself too.
5. One good and bad thing about dictionaries: You don't access them one by one sequentially. i.e., you cannot do eg[1] etc
6. How does python know if we want a dictionary object? (Example code: xyz = dict())
7. Use: one example is to build a word frequency list from a corpus.

How to work with dictionaries

DictionaryBasics.py

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

Let us do a small exercise

Exercise 3 in the Dictionaries Chapter
"Write a program to read through a mail log, build a histogram using a dictionary to count how many messages have come from each email address, and print the dictionary"

Dictionaries and Lists: questions

- What is the value of someThing after this statement?

```
someThing = {'a': 1,'b':2, 'a':3}
```

Dictionaries and Lists: questions

- What is the value of someThing after this statement?

```
someThing = {'a': 1,'b':2, 'a':3}
```

- What does this print?

```
a=[1,2,3,4,5,6,7,8,9]
a[None:6]
```

Dictionaries and Lists: questions

- What is the value of someThing after this statement?

```
someThing = {'a': 1,'b':2, 'a':3}
```

- What does this print?

```
a=[1,2,3,4,5,6,7,8,9]
a[None:6]
```

- What gets printed after the following two statements:

```
dict  = {"user","sowmya", "password","iastate"}
print(dict['sowmya'])
```

Dictionaries and Lists: questions

- What is the value of someThing after this statement?

```
someThing = {'a': 1,'b':2, 'a':3}
```

- What does this print?

```
a=[1,2,3,4,5,6,7,8,9]
a[None:6]
```

- What gets printed after the following two statements:

```
dict  = {"user","sowmya", "password","iastate"}
print(dict['sowmya'])
```

- What does this print:

```
numbers = [1, 2, 3, 4]
numbers.append([5,6,7,8])
print(len(numbers))
```

Tuples: An Introduction

- Tuples are a collection of items, that look very similar to lists in python.
- However, the main difference is that tuples are immutable, whereas lists are mutable.
- Dictionaries have a method called items(), that returns a list of "tuples", where each item is a key-value pair.

```
d = {'a':10, 'b':1, 'c':22}
t = d.items()
print(t)
[('a', 10), ('c', 22), ('b', 1)]
```

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

Tuples and Dictionaries

With tuples, we can have two iteration variables in a loop to read through a dictionary!

```
d = {'a':10, 'b':1, 'c':22}
for key, val in d.items():
    print(val, key)
#The output is:
10 a
22 c
1 b
```

Defining a Tuple

- t1 = 'a', 'b', 'c', 'd', 'e'
- t2 = ('a', 'b', 'c', 'd', 'e')
- t3 = tuple() #creates empty tuple
- t4 = ('a',)
- lst = [1,2,3]
- t5 = tuple(lst)

Tuple Operations

Several list and string operators work with tuples too. Let our example tuple be: t = ('a', 'b', 'c', 'd', 'e')

- t[0] gives us 'a'
- slicing operation t[1:3] gives us a tuple ('b', 'c')
- Something like: t[0] = 'A' will throw an error as tuples are immutable.
- But, we can change the whole tuple. This won't throw an error: t = ('A',) + t[1:]
- A very useful thing: (name,addr) = "sowmya@iastate.edu".split("@") - saves "sowmya" into name and "iastate.edu" in to addr.
- name,addr = addr,name swaps name and addr values in just one line!

Comparing Tuples

- Comparison of tuples (and also other sequences like lists) works in a slightly counter-intuitive way. E.g., (0, 1, 2000000) $<$ (0, 3, 4) will be true in Python.

Comparing Tuples

- Comparison of tuples (and also other sequences like lists) works in a slightly counter-intuitive way. E.g., (0, 1, 2000000) $<$ (0, 3, 4) will be true in Python.
- Sort example:

```
txt = 'but soft what light in yonder window breaks'
words = txt.split()
t = list()
for word in words:
    t.append((len(word), word))
t.sort(reverse=True)
res = list()
for length, word in t:
    res.append(word)
print(res)
```

Back to Tuples and Dictionaries

Sorting a dictionary by values.

```
d = {'a':10, 'b':1, 'c':22}
l = list()
for key, val in d.items():
    l.append((val, key))
print(l) #what will this print?
```

Back to Tuples and Dictionaries

Sorting a dictionary by values.

```
d = {'a':10, 'b':1, 'c':22}
l = list()
for key, val in d.items():
    l.append((val, key))
print(l) #what will this print?

[(10, 'a'), (22, 'c'), (1, 'b')]
l.sort(reverse=True)
print(l)
```

Tuples and Dictionaries -2

Using tuples as dictionary keys: This is useful when we want to have dictionary keys as having more than one element. An example is: writing a program to create telephone directory.. where you want to index by both firstname and lastname.

- Let us say we want to assign like this: directory[last,first] = number - (last,first) here is a tuple.
- To write a for loop for this dictionary, we then write: for last, first in dictionary: ...

How to work with all these data structures

- Lists, Dictionaries and Tuples can get very complex and confusing, especially if one is nested inside another. So, you should be careful while writing your programs and learn to effectively debug them.

How to work with all these data structures

- Lists, Dictionaries and Tuples can get very complex and confusing, especially if one is nested inside another. So, you should be careful while writing your programs and learn to effectively debug them.
- While debugging: when you are clueless about the errors you are getting, read the code line by line, work out manually for one or two examples to check if your program is doing everything right.

How to work with all these data structures

- Lists, Dictionaries and Tuples can get very complex and confusing, especially if one is nested inside another. So, you should be careful while writing your programs and learn to effectively debug them.
- While debugging: when you are clueless about the errors you are getting, read the code line by line, work out manually for one or two examples to check if your program is doing everything right.
- Use try-except blocks to catch known issues. Have print statements at necessary places during the debugging phase, to know what the code is doing.

How to work with all these data structures

- Lists, Dictionaries and Tuples can get very complex and confusing, especially if one is nested inside another. So, you should be careful while writing your programs and learn to effectively debug them.
- While debugging: when you are clueless about the errors you are getting, read the code line by line, work out manually for one or two examples to check if your program is doing everything right.
- Use try-except blocks to catch known issues. Have print statements at necessary places during the debugging phase, to know what the code is doing.
- Understand the different errors and why they occur.

How to work with all these data structures

- Lists, Dictionaries and Tuples can get very complex and confusing, especially if one is nested inside another. So, you should be careful while writing your programs and learn to effectively debug them.
- While debugging: when you are clueless about the errors you are getting, read the code line by line, work out manually for one or two examples to check if your program is doing everything right.
- Use try-except blocks to catch known issues. Have print statements at necessary places during the debugging phase, to know what the code is doing.
- Understand the different errors and why they occur.
- Keep experimenting and use online tutorials, search for advice on stackoverflow.com, look for video lectures.

Other Data Structures - 1

1. A Stack: Stack is a kind of data structure where we assume items as being stacked one above the other. So, whichever was put in last needs to be taken out first (this is called LIFO - Last in First Out). Example code and working: `http://goo.gl/g2W5pH`

Other Data Structures - 1

1. A Stack: Stack is a kind of data structure where we assume items as being stacked one above the other. So, whichever was put in last needs to be taken out first (this is called LIFO - Last in First Out). Example code and working: `http://goo.gl/g2W5pH`
2. A Queue: Queue is the reverse of Stack. It is a FIFO (First in First Out). Example code and working: `http://goo.gl/Yw4zB0`

Other Data Structures - 1

1. A Stack: Stack is a kind of data structure where we assume items as being stacked one above the other. So, whichever was put in last needs to be taken out first (this is called LIFO - Last in First Out). Example code and working: `http://goo.gl/g2W5pH`
2. A Queue: Queue is the reverse of Stack. It is a FIFO (First in First Out). Example code and working: `http://goo.gl/Yw4zB0`

Uses: internal implementation of search operations, performing arithmatic operations, recursive operations and many more. A good explanation of various ways to implement and apply stacks and queues is here: `https://goo.gl/2yA0OW`

Other Data Structures - 1

1. A Stack: Stack is a kind of data structure where we assume items as being stacked one above the other. So, whichever was put in last needs to be taken out first (this is called LIFO - Last in First Out). Example code and working: `http://goo.gl/g2W5pH`
2. A Queue: Queue is the reverse of Stack. It is a FIFO (First in First Out). Example code and working: `http://goo.gl/Yw4zB0`

Uses: internal implementation of search operations, performing arithmatic operations, recursive operations and many more. A good explanation of various ways to implement and apply stacks and queues is here: `https://goo.gl/2yA0OW`
Both a stack and a queue can be implemented using the list data structure in Python. Try this out by writing your own code (call them Stack.py and Queue.py) for these.

Other Data Structures - 2

1. A Tree: A tree is a data structure with nodes and connections between them, arranged in a root, branch, leaves style.
   - use: representing hierarchical information, storing data for searching through, for routing information from one location to another.
   - Trees also can be implemented using Lists in Python. Example: `http://goo.gl/UYE4yD`

Other Data Structures - 2

1. A Tree: A tree is a data structure with nodes and connections between them, arranged in a root, branch, leaves style.
   - use: representing hierarchical information, storing data for searching through, for routing information from one location to another.
   - Trees also can be implemented using Lists in Python. Example: `http://goo.gl/UYE4yD`
2. A Graph: This has a set of points called nodes or vertices and a lot of connections between them.
   - use: any kind of problem which involves a network of relations (e.g., analysing social network, molecule interactions, computer networks finding the shortest route etc.,)
   - Graphs also can be implemented in Python using dictionaries and lists. Example: `http://goo.gl/3Tkjh2`

Exercise for today: post solutions later

- Exercise 2 in textbook chapter: "Write a program that categorizes each mail message by which day of the week the commit was done. To do this look for lines that start with "From", then look for the third word and keep a running count of each of the days of the week. At the end of the program print out the contents of your dictionary (order does not matter)."
- Note: You can use mbox-short.txt as the file to estimate these numbers.
- Hints: We need knowledge of- file processing, regular expressions, lists, dictionaries, strings to solve this.

Next Week

- Tuesday: Review and Practice So, post any questions you have online in the forum with "Recap: 27 Feb" as the title.
- Tuesday: I will prepare some exercises that will help you with Assignment 4
- Thursday: Basics about making small applications that work on browsers.
- Tasks for you: Revise and have questions. Post questions on forum by Monday.
- Browse through this website: `https://bottlepy.org/docs/dev/`

Some Pythonic Fun

"30 Python language features and tricks you may not know about"
`http://sahandsaba.com/thirty-python-language-features-and-tricks-you-may-not-kno`
`html`