Basics of Programming

L09: Tuples

Apr 2020

Dr. Ram P Rustagi
Professor, CSE Dept
KRP, KSGI
rprustagi@ksit.edu.in

Resources and Acknowledgements

- Python for everybody
  - https://www.py4e.com
- A first course in programming
  - https://introcs.cs.princeton.edu/python/20functions/
- netacad.com: Python Essentials:

https://780671818.netacad.com/courses/1004579/modules/items/66720226

Collections

- Collection:
  - Multiple values together in a single entity
  - Multiple values are represented by a variable
  - Allows checking for existing of a value
  - Allows adding new value, removing existing value
  - Allows modification/update of existing values
- Collections studies so far:
  - Lists: values are accessed by index
  - Dictionary: elements accessed by key/label
- New collection type: Tuples
  - Very much like Lists, but immutable

Tuples

- A collection (sequence) of elements which functions like a **List**.
  - Elements are indexed start from 0
  - Elements are enclosed in parenthesis
- Examples

```
vowels=(‘a’, ‘e’, ‘i’, ‘o’, ‘u’)
for i in range(len(vowels)):
  print (vowel[i])

for elem in vowels:
  print (elem)
```

Tuples: Immutability

- Tuples are immutable
  - Element at ANY index can’t be modified
  - Can not sort inplace, append to it
- Examples

```
alphabets = (‘a’,’b’,’c’,’d’,’e’)
alphabets[0] = ‘A’
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item
assignment
>>> alphabets.append(‘f’)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute
'append'
```

Tuples vs. List

```
vowels=(‘a’, ‘e’, ‘i’, ‘o’, ‘u’)
dir(vowels)
   [‘count', 'index']
```

```
vowlist =[‘a’, ‘e’, ‘i’, ‘o’, ‘u’]
dir(vowlist)
[‘append', 'clear', 'copy', 'count', 'extend',
'index', 'insert', 'pop', 'remove', 'reverse',
‘sort']
```

- Tuples are more efficient that lists
  - In terms of memory and performance
  - python does not have overhead of modification
- Whenever using temporary variable
  - preferably use tuples (if immutable)

Tuple Operations

- Like lists, elements can be of different type
- Tuple with a single element

```
x= (2,)
```

- Tuples can be used for assignment on the left side of an statement, e.g.

```
(geom,x,y) = (‘coordinate’,2*2, 3*3)
print(geom)
print(x)
print(y)
```

- Comparing tuples

```
(1,2,5,6) < (1,2,6)
(1,2.5) < (1.1,5)
(1,2) > (1.0,)
```

Tuple Operations

- Tuple packing
  - `x= (1, 4, 9)`
  - `print(x[1])`
- Tuple unpacking
  - `a,b,c = x`
  - `print(b,c)`

Tuples and Dictionaries

- Dictionary mehod items() returns tuples of (key,value)

```
ascii={}
ascii[‘A’] = 65
ascii[‘B’] = 66
ascii[‘a’] = 97
for (key,value) in ascii.items():
  print(key, value)
#
tups = ascii.items()
```

Sorted: Dictionary Tuples

```
ascii={}
ascii[‘A’] = 65
ascii[‘a’] = 97
ascii[‘b’] = 98
ascii[‘B’] = 66
```

- `sorted()` is a built-in function

```
sorted(ascii.items()) #sorts by keys
 [('A', 65), ('B', 66), ('a', 97), ('b', 98)]
```

- Sort by values instead of keys

```
t=[]
for k,v in ascii.items():
  t.append((v,k))
sorted(t)
sorted(t, reverse=True)
```

Word Count Program

- Find top 5 words occurring in a file of text

```
import sys

filename = sys.argv[1]
fh = open(filename)
count={}
for line in fh:
  words = line.strip().split()
  for word in words:
    count[word] = count.get(word,0) + 1

wlist = []
for k,v in count.items():
  wlist.append((v,k))
slist = sorted(wlist, reverse=True)
print(slist[:5])
```

Top words: List Comprehension

```
import sys

filename = sys.argv[1]
fh = open(filename)
count={}
for line in fh:
  words = line.strip().split()
  for word in words:
    count[word] = count.get(word,0) + 1

wlist = [(v,k) for k,v in count.items()]
print(sorted(wlist, reverse=True)[:5])

```

Summary

- Tuple syntax
- Single element tuple require a comma
- Tuples are immutable
- Tuples are comparable
- Tuples in assignment (on left side)
- Sorting dictionary by key or values
- List comprehension method

Summary So Far

- Discussed so far (Basics of python)
  - Variables, and primitive data types
  - Turtle Graphics
  - Control stuctures
  - Loops (for and while)
  - Collections
  - Lists, Dictionaries, Tuples
  - General Exercises
- Next lectures
  - Recursion (a powerful programming technique)
  - Network programming (Using HTTP), web services
  - Data visualization
- Python Advanced (may not be covering)
  - OOPS, Connecting to DB, lamda functions etc.

Exercises

- Ex01: Three sum problem
- Given a number of integers (+ve and -ve) in a file, integer values separated by comma in a line,
- write a program that prints all tuples whose sum of elements is zero.
- For example if file contents are

```
8,30,-30,-20,-10,40,0,10,5
```

Answers

```
(30, -30, 0)
(30, -20, -10)
(-30, -10, 40)
(30, -30, 0)
(-10, 0, 10)
```

Exercise 2

- Dictionary of tuples:
  - For all of your subjects studied last year
  - create a dictionary with subject as the key and tuple of marks as the value
- Your marks are tuples
  - Marks once obtained, don’t change, hence immutable
  - Tuple having 3 elements
    - Internal marks
    - Main exam marks
    - Total marks

Exercise 3

- Mini project: Build a Tic-Tac-Toe game
- Game has nine positions as below. P

```
1 2 3
4 5 6
7 8 9
```

- User input move is marked as X,
- Your program's move is marked as O
- For example, user input=5, program move=9

```
1 2 3
4 X 6
7 8 O
```

- At each step ask user to move, put your move and display the result if user won or lost.

Exercise 04

- Do the following quizzes in netacad
  - `3.2.1.1`
  - `4.2.1.1`

Questions