WEEK 2 (CONTINUED)

Lists, Tuples, Dictionaries

Python builtin types

- How do you declare the following:

- List
- Tuple

- What are the differences between lists and tuples?

Sequences

- String
  - Unicode string
  - Byte array
- Tuples
  - Myfirsttuple = (‘something’, ‘thing2’)
- Lists
  - Mylist = [‘something’, ‘thing2’]

- Do they have similar methods/functions?

Strings are Sequences

- **String**: a sequence of characters
- Strings are surrounded by double (“) or single (‘) quotes
- Are **immutable**; can be combined (concatenated); sliced
- Methods: **capitalize, upper, lower, count, find, replace, endswith, isspace**
- Handling long strings
- String multiplication
- File paths as strings

Using Sequences

- Indexed collections of values
  - strings, lists, tuples
  - “**0**”-indexed from left
  - Indices start at “0”
  - “**-1**”-indexed from right

Common operations:

- Getting an element of a sequence `[i]`
- Getting a range of elements in a sequence `[i:j]`

Sequence splicing

- In all sequences “:” -- means ALL
- Python sequences are up to, but not included.
  - “my python class”[0:2]
  - “my”

Tuples

- Immutable Sequence
  - No Slicing or Dicing
    - Mytuple = (1,6,9,3)
    - Mytuple(:3)
- Index
- Count

Lists

- Mutable Sequences
- Flexible container objects
- Hold any objects; or be empty
- Can be retrieved in parts (slicing)
- The range(args) function
- Can be added, multiplied
- Functions
  - Membership
  - Append, Extend, Insert, Remove, Reverse, Sort

Slicing

- Start
- Stop
- Copy everything
- Count backwards
- Step
- Membership (in)

CODE IT

1. Write a program with the following functionality
    1. Print all the elements of list
    2. Print the elements of the list in reverse order
    3. Print only element number 4
2. Write a program with the following functionality:
    1. Create a list with multiple types assigned to different elements
        1. Testlist = [‘sometext’, 5 , ‘morewords’ ,3, 8 ,1,2]
    2. Cycle through the elements in the list if type is string, print the number of letters.
    3. If the type is integer, see if it is even.
    4. If it is float, multiply is by
3. Write a program that prompt the user to enter text. Then print the sentence framed in a box.

Dictionaries: {key : value}

- Hash (mapping) tables
- Consist of **items** pairs of **key : value**
- Accommodates heterogeneous content
- Key are immutable and unique
- Values are mutable
- Helpful mini-databases
- Key assignment
- Functions: has_key, keys, values, items, get, fromkeys, deepcopy (from copy module), pop

Dictionary key

- Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.
- Keys are **unique** within a dictionary while values may not be. The values of a dictionary can be of any type, but the **keys must be of an immutable data type such as strings, numbers, or tuples**

CODE IT

- Make a dictionary
  - HaynesKidsAge ={‘David’: 3, ‘Olivia’: 1}
- Use key to retrieve value, How old is Olivia?
  - HaynesKidsAge.has_key(‘aname’)
  - HaynesKidsAge[‘Olivia’]
- Add new entry to dictionary, New child Tim
  - HaynesKidsAge[‘key’] = value
- David had a birthday how old update your dictionary
- What is the range of ages of the kids?
  - Retrieve all values from dictionary

CODE IT

- Dictionary.get
  - People.get(‘Sam’, ‘Sam is not your child’)
  - Returns textual statement

Dictionary String formatting

```
myDict = {“test”: “David”}
print("Something goes here {test}
".format(**myDict))
```