WEEK 2

Conditionals, Lists, Tuple, Dictionaries

CONDITIONS: BOOLEAN

Expression that is either true or false.

“False “ in Python:

1. False
2. None
3. 0
4. “”
5. ()
6. []
7. {}

BOOLEAN EVALUATION

== is a comparison operator

others are:

`x != y`	`x is not equal to y`
`x > y`	`x is greater than y`
`x < y`	`x is less than y`
`x >= y`	`x is greater than or equal to y`
`x <= y`	`x is less than or equal to y`

= is the assignment operator

LOGICAL OPERATORS

Three logical operators: and, or, and not.

- Meaning of these operators similar to their meaning in English.

  For example,

  ```
  x > 0 and x < 10
  ```

  is true only if x is greater than 0 and less than 10 (if both conditions are true

CONDITIONAL STATEMENTS

- Conditional statement: if…elif…else

```
if x == 1:
    print “x is 1”
elif x == 2:
    print “x is 2”
else:
    print “x is not 1 or 2”
```

- **colons** are used at the end of each condition (:)
- **indentation** defines what executes for each condition

CODE IT UP!

- `d = 0`
- `if d == False: print(‘David doesn’t know what he is doing again!’)`
- `if d == True: print(‘Well maybe he does know a few things’)`

- Write a program that prompts the user to input a string.
  - Statement = input(‘Please enter something: ’).
  - If the users input a value, print out the input statement.
  - If there is nothing input, print.
    - You didn’t enter anything.

SEQUENCES

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

- Have similar methods/functions

STRINGS ARE SEQUENCES

- **String**: a sequence of characters
- Strings are surrounded by double (“), single (‘) or triple 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; “`-1`”-indexed from right

Common operations:

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

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
  - `kidsAge ={‘David’: 3, ‘Olivia’: 1}`
- Use key to retrieve value, How old is Olivia?
  - `kidsAge.has_key(‘aname’)`
  - `kidsAge[‘Olivia’]`
- Add new entry to dictionary, New child Tim
  - `kidsAge[‘Oscar’] = 25`
- David had a birthday how do you 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))`