Data structures

String

Methods

- .lower() returns the string with all lowercase characters.
- .upper() returns the string with all uppercase characters.
- .title() returns the string in title case, which means the first letter of each word is capitalized.
- .split() or .split(delimiter) returns an array of substrings split by the delimiter
  - delimiter by default is spacebar, but it could be any string or '\n' or '\t' etc.
- .join() is the reverse of split, it joins strings together
  - Syntax is : 'delimiter'.join(listToJoin)
  - where delimiter is the filler to add between the array elements
- .strip() or strip('delimiter') removes the delimiter from the start and end of a string
- .replace(subStringToBeReplaced, newSubString)
  - string = (Hello World and stuff)
  - string.replace(' ', '__')
  - string is now Hello_World_and_stuff)
- .find(subString) finds the first index value where that string is located
- .format() adds variables to a string
  - "My fav { } is { }."format(category,item)
  - "My fav {category} is {item}."format(category=category,item=item)

Escape characters

In order to include for instance quotations marks in a string just add a backslash before it

String + Variable

In general you should use the method .format() to do this
but to to manually do so you can do:

coolString = "This is my "+ str(variableName)+" Cool string"

Multi Line String

If you want to make a string span multiple lines you use a triple comma operator to start and end the string

Example

```
leaves_of_grass = """
Poets to come! orators, singers, musicians to come!
Not to-day is to justify me and answer what I am for,
But you, a new brood, native, athletic, continental, greater than
  before known,
Arouse! for you must justify me.
"""
```

Print(String, Int)

I can't print out like this:
print("Some String"+int)

I have to do it like this:
print("Some String", int)

Related to:

[[Python]] #Python #Syntax #String

"x" in "exist"

print("e" in "blueberry")
=> True

print("a" in "blueberry")
=> False

List

myList = [ ]

new concepts

- Lists can have multiple types in the same list
- 1 selects the last element in a list, (-3 would select the 3rd last element of the list)
- mySlicedList = myList[2:10] Returns a list from index 2 up until (excluding) index 10
  - myList[:n] Returns the first n elements of the list
  - myList[-n:] Returns the last n elements of the list
  - myList[:-n] Returns all but the last n elements of the list
- List Comprehension
  - numbers = [2, -1, 79, 33, -45]
  - doubled = [num * 2 for num in numbers]
  - add = [num + 2 for num in numbers if num < 0 else num + 3]

List Methods

.insert(index, element)

.pop() OR pop(index)

.sort() = sorts the **current** list
If you want to sort largest to smallest = .sort(reverse=true)

sorted(myList) = returns a **new** sorted list

.count("myElement") = Number of occurrences of an element in the list

len(myList) = returns a list length

Related to:

[[Python]] #Lists

Tuple

a tuple is very similar to a list, but it's immutable (Final in java)

my_tuple = ("myString", 24, "somethingElse")
Name, age, jobTitle = my_tuple // This line will save the values of my_tuple into these new variables

Edge case: if I need to make a 1 element tuple I'll need to do (24,) Otherwise it'll just think the brackets are order of operation brackets

Dictionary

A dictionary is an unordered set of key: value pairs.

- myDictionary = {"avocado toast": 6, "carrot juice": 5, "blueberry muffin": 2}
- myEmptyDictionary = {}
- Add a key: myEmptyDictionary[key] = value
  - We can overwrite key:value pairs the same way
- .update() is used If we wanted to add multiple key : value pairs at once
  - myEmptyDictionary.update({"toast": 3, "juice": 9, "muffins": 200})
- Access data with: myDictionary.get(key)
  - myDictionary.get(key, defaultIfItDoesntExist)
- .pop(key:defaultIfItDoesntExist) removes a pairing from the dictionary
- .keys() - returns an object of key's in the array
- .values() - returns an object of value's in the array
- .items() - returns (key,value)