**Python Notes**

# ## Python DATA STRUCTURES ##

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

# PYTHON | C

# list = arrays

# tuple = arrays[][]- typically reserved for coordinate pairs.

# dict = map{key: value}

# set = unordered\_set()

# set items are unordered, unchangeable, and do not allow duplicate values. Like arrays with additional built-in functionality

# range = for loops = range(i) - i is the number of times through the loop starting at 0 index.

import csv

import sys

import random

# Number of simluations to run

N = 1000

# Compiling isn't a separate step in Python

# So, if you make a syntax error within a func def, but it isn't called in main, it won't register as a problem.

# Be sure to check code frequently in python when adding func()s

### MAIN ###

def main():

...

# funWithLists()

# funWithDicts()

# funWithSets()

# funWithTuples()

####### TUPLE #########

# tuples are immutable collection of objects | const lists[]

# No add() or .insert() function

def funWithTuples():

...

\_tuple = ('1', '2', '3')

print("Original Tuple: ", \_tuple)

\_tuple2 = \_tuple + ('4',) # by using an Original Tuple and adding one value (don't forget comma!)

print("New Tuple:", \_tuple2)

tuple1 = ("abc", 34, True, 40, "male") # tuples can have multiple variable types

print(tuple1)

# tuple1.add("female") | ERROR - Won't compile - No add(), update(), or insert() functions

# tuples are constant | Need to be initialized with values.

####### DICT / MAPs ######

# DICT() VALUES in a dictionary can be of any data type and CAN be duplicated

# DICT() KEYS CANNOT be repeated and must be immutable

def funWithDicts():

...

myDict = {}

ryan = {"Ryan": "4", "RyanJ": "5"} # separate dictionary from myDict

# bob = {"Bob", "2"} - WRONG, need a colon, not a comma here

tyler = {"Tyler": "3"} # separate dictionary from myDict

# myDict.add(ryan) | WRONG - won't compile

# myDict.add(tyler) | WRONG - won't compile

myDict["Ryan"] = 1 # | CORRECT

myDict["Bob"] = 2

myDict["Tyler"] = 3

myDict[4] = 5

myDict[6] = "six"

myDict[7.7] = "float"

print()

print("-------------Original Dictionary---------------")

print(myDict)

print()

print("-------------Combining Dictionaries---------------")

# There is NO add(), append(), or insert() method you can use to add an item to a dictionary in Python.

# Add an item to a dictionary by inserting a new index key into the dictionary, then assigning it a particular value.

# Ex. myDict["Ryan"] = 1

# Elements are NOT created separately and then added to Dictionary!

### Combining dictionaries - doesn't really work, again, because this isn't an array, you aren't indexing, you are mapping.

myDict["newDictionary"] = ryan # adds an additional key ("newDictionary") and then the value becomes the original key-value pair.

print("myDict:", myDict) # {'newDictionary': {'Ryan': '1'}} - key-value pair added to end of dictionary - additional key is confusing

print()

print("------ OVERRIDING key 'newDictionary' with new value-------")

myDict["newDictionary"] = tyler # tyler will override previous key value of 'name' - can't have two identical keys

print("myDict:", myDict)

print()

print("---------- Printing Just Keys--------------")

for key in myDict.keys():

print(key)

print()

print("---------- Printing Just Values--------------")

for value in myDict.values():

print(value)

print()

print("-------- Printing Both Key-Value Pairs (Items)-------")

for item in myDict.items():

print(item)

print()

####### SETS ########

# Sets can contain multiple variable types, but they are unordered.

# Cannot insert or delete specific elements using indexes

# https://www.geeksforgeeks.org/sets-in-python/

def funWithSets():

empty\_set = set() # create an empty set

# check data type of empty\_set

print('Data type of empty\_set:', type(empty\_set))

empty\_set = {118, 114, 116, 113, 115}

print(empty\_set) # set() auto arranges values for ints

for x in empty\_set:

print(x)

empty\_set.add(125)

empty\_set.add(1452)

print("------- 2 Values added manually with .add() function ----------")

print(empty\_set)

secondSet = {109, 114, 122, 125, 1453}

print("-------- Second Set -------------")

print(secondSet)

empty\_set.update(secondSet)

print("------------------------------------------------------")

print("Updating initial empty\_set with a 2nd set with empty\_set.update(secondSet))")

print("------------------------------------------------------")

print(empty\_set)

thirdSet = {108, True, int("123"), 0, False, "Harry", "Dick", "Sally", 43, 55, 67}

print("------------------------------------------------------")

print("Third set with random variable types")

print("------------------------------------------------------")

print(thirdSet)

print("------------------------------------------------------")

empty\_set.update(thirdSet)

print("Updating initial empty\_set a 2nd time with a 3rd set of different variable types")

print(empty\_set)

# set using constructor syntax

thisset = set(("apple", "banana", "cherry")) # note the double round-brackets # don't need to us this syntax

print(thisset)

# myset[1] = "Hello" | ERROR - cannot assign indexes to sets because they are unordered

# print(myset)

####### LISTS ######

def funWithLists():

myList = []

myList.append(3)

myList.append(4)

myList.append(5)

myList.append(6)

myList.append("seven")

print(myList)

print("-----------------------------")

## BAD ## | WONT COMPILE

# for i in range:

# print(myList)

listSize = len(myList) # - length of array

for i in range(listSize): # passes the length of the array into range function | range needs an int value

print(myList)

print("-----------------------------")

for i in myList: ## didn't use range here - Lines 41 and 46 are the same, different ways to find how many times to iterate.

print(myList) ## prints entire list of elements on one line - for each element in array, so 3 times here

print("-----------------------------")

## BAD ##

# for i in myList:

# print(myList[i])

# Don't pass an indexed value into print unless iterating with range

# for element creates a variable element starting at index 0 of myList container

# in myList essentially finds the size of myList, and iterates over myList, creating a new variable one index at a time

for i in myList:

print(i)

print("-----------------------------")

for i in range(len(myList)):

print(i) # only prints index points of myList ## because we used range here

print("-----------------------------")

for i in range(len(myList)):

print(myList[i])

## BOTTOM line - Use indexed values when using range function,

# otherwise just do something to variable created in for loop.

# main() call

if \_\_name\_\_ == "\_\_main\_\_":

main()

"""

List is a collection which is ordered and changeable. Allows duplicate members.

Tuple is a collection which is ordered and unchangeable. Allows duplicate members.

Set is a collection which is unordered, unchangeable\*, and unindexed. No duplicate members.

Dictionary is a collection which is ordered\*\* and changeable. No duplicate members.

"""