Wk	Focus & Medium	Weekly Topic & Assignment
7 Oct 10 to 15	(1 of 4)	

QUIZ Instructions
QUIZ Answer

Objective: more exercises on python pillars to prepare for creating an object generator.

We will review in class but you will need to answer and turn it in when finished. Turn it in by the 19th the latest but won't take you long.

- I will post everyone's own gradebook this week.

- The 2nd part of the week will review class objects

versus

```
"""# -*- coding: utf-8 -*-
Created on Mon Oct 10 10:59:53 2022
@author: 17574
====================
#==========================
#=============================================================
#=> it.304 2nd Graded Assignment
#=============================================================
#========================="""
import pandas as pd              #dataframe library
import numpy as np               #numeric library
import matplotlib.pyplot as plt  #visualization library
import os
os.chdir('c:\\Users\\17574\\Desktop\\data') #microsoft uses 2 \\
df0 = pd.DataFrame() #explicitly set the data object
df0 = pd.read_excel("shakes_corpus_v1.xlsx") #ETL method 2
df0.info()
mydict = df0.to_dict()
#====================================
#=>1.0 Pillar: Iterators
'''1.1 Task: use an iterator and produce total words all plays'''
#====================================

#==> ENTER YOUR CODE HERE
mylist = []
for i in mydict['script'].values():
    mylist.append(i)
total_script_characters= 0    #how many characters?
for i in mylist:
    total_script_characters = total_script_characters + len(i)
total_script_characters

# Answer: 1,212,379

'''1.2 Task: what is easiest in code to double total characters'''
#==> ENTER YOUR CODE HERE

total_script_characters*2

# Answer: 2424758

#====================================
#=> 2.0 Pillar: Functions
'''Task: Generate a tuple wth the code provided
   hint: use codebook '''
#====================================
mylist = []
mytuple = ()
for i in range(37):
    mylist.append(i)

#==> ENTER YOUR CODE HERE
mytuple = tuple(mylist)

# Answer:
print(mytuple)         # (0,1,..............36)
print(type(mytuple))   # tuple
```

7 Oct 10 to 15	(2 of 4) **versus** **missing Danny and Jackson memes**	see code below

```
#========================================
#=> 3.0 Pillar: Built-in objects - Sets
#=========================================
''' 3.1 Quickly explain what this statement is doing

    random.randint(len(mydict),len(df0['script']))

    3.2 What does the type() function tell you and why is it
        important?

    3.3 Create one set from =mydata1 and mydata2
    3.2 Use the type() function to prove it is a set
    3.5 Why is performing housekeeping a good habit?'''
#====================================

import random  # generates random numbers
               # randint(start value, end value)
mydata1 = random.randint(len(mydict),len(df0['script']))
print(len(mydict),len(df0['script'])) #4, 37

#==> 3.1 ENTER YOUR RESPONSE HERE
'''pulling random value from 4 to 37'''

#==> 3.2 ENTER YOUR RESPONSE here after the 3 lines of code
type(mydata1)
mydata1 = (mydata1,)
type(mydata1)

'''can only add objects that are the same object type'''

#==> 3.2 ENTER YOUR RESPONSE HERE
mydata2 = 1,2,3,4,3,2,1
myset = set(mydata1 + mydata2)

#...ANSWERS:
#Answer: <your code answers should be the same except m
       #each person will have 1 diff value
print(mydata1,set(mydata2))     # 35, {1,2,3,4}
print(myset)                    # {1, 2, 3, 35, 4}
print(len(myset))   # 3.1 => 4
print(type(myset))  # <class 'set'>

#Answer built in objects only take one parameter.
# BUT you can add objects together as long as they are the same
# object type.

# housekeeping
#Why: so dont absob data you dont need later by accident
del mydata1; del mydata2;del myset

#====================================
#= 4.0 Pillar - interpreting packed built-in objects
'''Task: you have the following object visible to your in your
   'variable explorer' window. if script is in the ... describe
   the object container around it and what you would do to
    unpack it.'''
#====================================
'''             [({...})],   '''
#====================================
the string data is in a dictionary
which is inside a tuple
which is inside a list
```

Wk	Focus & Medium	Weekly Topic & Assignment
7 Oct 10 to 15	3 of 4) classes! this is is not using the self parameter so functions are outside of the object	

```
"""# -*- coding: utf-8 -*-Created on Mon Oct 12 10:59:53 2022
@author:17574 b.hogan@snhu.edu it.304.fall.22
# WEEK 7 CODE final - including classes """
                        #==========================================
                        #=>week 7  Object Classes Overview
                                #====================================


Lexical Analysis
        always remember about indent \ dedent!
        if you copy and paste and teh spacing is wrong it wont run

https://python.readthedocs.io/en/latest/reference/lexical_analysis.html

#Create a report structure
mydict = {"training done":[], "total animals":0}
class myFarm:    #create parent class object
    pass
    name = ""
    species = ""
    train = ""
def add_train(traintype):    #create a user function to count, sort
    mydict["training done"].append(traintype)
    mydict["total animals"] =+1

#------------->   #children instantiate from parents
a1 = myFarm()     # instantiate children objects from parent, a for animal
a2 = myFarm()     # all object names are user defined

#update attributes
a1.name = 'mackenzie'  #object.attribute or object.function
a1.species ='dog'
a1.train = 'speak'
add_train(a1.train) #cheCK-OUT!   <only here bc space>

a2.name = 'vinny'
a2.species = 'horse'
a2.train = 'jumping'

add_train(a2.train) #'''function accepts attribute to update dictionary
object'''

#write a simple report using a dictionary data object format
mydict_rpt = {a1.name:a1.species, a2.name:a2.species,"metrics=>":mydict}
mydict_rpt
 '''{'arnold': 'dog','vinny': 'horse','metrics=>':
     {'training done': ['catch', 'jumping'], 'total animals': 1}}'''

#use object's constructors to view its contents
print(a1.__dict__,a2.__dict__)
   ''' {'name': 'arnold', 'species': 'dog', 'train': 'catch'}
     {'name': 'vinny', 'species': 'horse', 'train': 'jumping'}'''

dir(a1)
['__class__', '__delattr__', '__dict__','__dir__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__',
 '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__',
 '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
 '__repr__','__setattr__', '__sizeof__','__str__', '__subclasshook__',
 '__weakref__',
 'name',
 'species',  'train']
```

7	Oct 10 to 15	

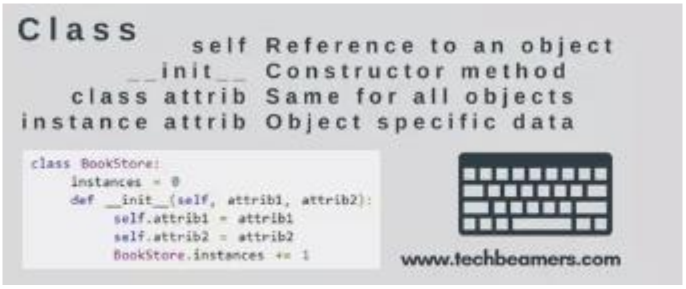
```
#============================================================
                        #=>Week 7 Objects part II
        #============================================================

#==> this is using programmming construct of .self.

class dog_train:
    name = ""
    num_fetch_train = 30
    num_fetched = num_fetch_train
    trainer_ok = 0

    def fetch_train(self, num_balls):
        self.num_fetched = self.num_fetched - num_balls
        if self.trainer_ok == 0 and self.num_fetched <= 0:
            return "sorry! {} not fetch trained. {} balls over a target of
{}".format(self.name,abs(self.num_fetched),self.num_fetch_train)
        elif self.trainer_ok == 1:
            return "Whew! {} passes training after {} balls".format(self.name, abs(self.num_fetch_train-
self.num_fetched-1))
        else:
            return"{} on target to pass fetch train with {} balls
left".format(self.name,self.num_fetch_train-self.num_fetched)

dog1 = dog_train()
dog1.name = "cheeseman"
print(dog1.fetch_train(9))
print(dog1.fetch_train(31))
dog1.trainer_ok = 1
print(dog1.fetch_train(1))
==============================
```



Class

self	Reference to an object
__init__	Constructor method
class attrib	Same for all objects
instance attrib	Object specific data

```
class BookStore:
    instances = 0
    def __init__(self, attrib1, attrib2):
        self.attrib1 = attrib1
        self.attrib2 = attrib2
        BookStore.instances += 1
```

www.techbeamers.com

Class, object, and function definitions:

Classes – are a framework or template for creating objects, attributes, and methods.

Objects – are the actors performing work. Child objects instantiate from parent objects and may contain their attributes and methods or have distinct attributes and methods.

Methods - are object instructions detailing how to perform behaviors in a class such as data arrangement, computation, printing, and conditional logic trees, perhaps to test, parse, or look for specific information. Methods do not have to return a value!

Functions – a set of instructions to accomplish a task independent of an object and typically part of a program. They may accept arguments and always return a value.

Class attributes – user-defined names that describe features of a class, and methods can use their values. For example, an object's unique ID, color, name, or numeric value for use in a calculation.

.self <self.attribute> is the first argument in a class function identifying its own attributes.

Essential Python tools associated with objects.,

Built-in types – Python core boolean, comparators, numeric types, and operations like 1+1, iterator types, and operations. REVIEW recommended!

Python Essential Data structures – lists, tuples, sets, dictionary, looping, more on conditionals. Methods and tips and tricks.