# Project 5

# Section: CPE101

# Name: Claire Minahan

# Instructor: S. Einakian

crimes.py template

#Crime class

class Crime:

#constructor

def \_\_init\_\_(self, crime\_id, crime\_category):

* set self.id equal to the crime\_id
* set the self.category equal to crime\_category
* set the self.day\_of\_week equal to nothing
* set the self.month equal to nothing
* set the self.hour equal to nothing

#boilerpoints

def \_\_eq\_\_(self,other):

* compare the type of self and other
* compare the id of self and other
* return True if both comparisons above return true, False otherwise

def \_\_repr\_\_(self):

* set representation for printing the attributes of the class

def \_\_str\_\_(self):

* turn the object into string format

# change the time components to proper format

def set\_time(self,day\_of\_week, month, hour):

* set the self.day\_of\_week equal to day\_of\_week
* check what the month parameter is equal to (string num)
* set the self.month equal to the month that corresponds with the month parameter
* check the hour parameter and set the self.hour equal to the hour parameter

# swap two variables in a list

# list int int 🡪 none

def swap(A, X, Y):

* set a temporary variable equal to a index in list A
* set list index X in A equal to list index Y in A
* set Y in A equal to the temporary variable

# access the crimes file and turn it into a list

# none 🡪 list

def crimes\_tsv():

* open the crimes.tsv file to read
* set empty list and x to 0
* for loop through the file
* disregard the first line
* turn the line into a list
* set id variable equal to the first index in the line
* set list2 to contain id and category
* append list2 to list1
* close crimes.tsv

# access the times.tsv file and return a list of all the crimes

# none 🡪 list

def times\_tsv():

* open the times.tsv file to read
* set the empty list and x to 0
* for loop through the file
* disregard the first line
* turn the line into a list
* set id variable equal to the first index in the line
* set date variable to the second index
* set time variable to the third index
* set list2 to contain id and date and time
* append list2 to list1
* close times.tsv

# create a list with no duplicates of only robberies and put into numerical order

# list 🡪 list

def create\_crimes(lines):

* make a list of only Robberies
* sort through and check for equality between ids
* take out any duplicate ids in list
* sort through the list and put into numerical order by id
* return the new list

test\_list = [[234235, ‘ROBBERY’], [238523, ‘FIRE’], [234234, ‘ROBBERY’], [2382934, ‘ROBBERY’]]

result\_list = [[234234, ‘ROBBERY’], [234235, ‘ROBBERY’], [238934, ‘ROBBERY’]]

self.assertListAlmostEqual(create\_crimes(test\_list), result\_list)

# returns the crime objects matching the given ID

# list int 🡪 int

def find\_crimes(crimes, crime\_id):

* for loop from 0 to length of crimes list
* if the current object in list id is equal to the crime\_id, return index

test\_list = [[234234, 'ROBBERY'], [234235, 'ROBBERY'], [2382934, 'ROBBERY']]

self.assertEqual(find\_crimes(test\_list, 234235), 1)

# add the time components to their crime object

# list list 🡪 list

def update\_crimes(crimes, lines):

* for loop from 0 to length of line list
* use find\_crimes to find index of id from lines
* add the time attributes to the index in crimes

test1 = [[2382374, 'ROBBERY'], [234235, 'ROBBERY'], [2382934, 'ROBBERY']]

test2 = [[2382374, 'Tuesday', '01/06/2015', '12:35'], [234235, 'Saturday', '12/13/2017', '03:18'], [2382934, 'Thursday', '05/23/2011', '05:19']]

result1 = [[2382374, 'ROBBERY', 'Tuesday', '01/06/2015', '12:35'], [234235, 'ROBBERY', 'Saturday', '12/13/2017', '03:18'], [2382934, 'ROBBERY', 'Thursday', '05/23/2011', '05:19']]

self.assertListAlmostEqual(update\_crimes(test1, test2), result1)

# find the most popular dates of robberies

# list 🡪 list

def count\_crimes(crimes):

* set total\_robberies to length of crimes list
* open robberies.tsv file to write
* write a header into file
* create Crime objects of each line taking in the first and second indexes
* use set\_time function from Crime class to put into proper format
* turn objects into strings with \_\_str\_\_ and write to the file
* loop through list and add 1 for each occurrence of the day, month, and time
* put each into a list and find the one with the maximum number of occurrences
* return the max number of occurrences for each attribute of the object

test1 = [[234234, 'ROBBERY', 'Saturday', '12', '03'], [234235, 'ROBBERY', 'Saturday', '12', '15'],[2382934, 'ROBBERY', 'Thursday', '05', '15']]

result1 = 'NUMBER OF PROCESSED ROBBERIES: 3 \nDAY WITH MOST ROBBERIES: Saturday\nMONTH WITH MOST ROBBERIES: December\nHOUR WITH MOST ROBBERIES: 3PM'

self.assertAlmostEqual(count\_crimes(test1), result1)

# put all the functions together to run

# none 🡪 string

def main():

* run the crimes\_tsv() function and set equal to crimes
* run the times\_tsv() function and set equal to times
* run create\_crimes(crimes, times) and set equal to crimes1
* run update\_crimes(crimes1, times) and set equal to updateCrimes
* print the count\_crimes(updateCrimes) function 54