Documentation for Assignment 1 - Python

Eric Sobczak

Answers to Assignment Questions

(a) What is an IDE?

> IDE stands for integrated development environment. It is what enables us to more easily write and execute code. Specifically for Python and Arduino in our situation. IDEs will provide features like code color coding and error detection. More advanced IDEs will auto fill code and make it easier to find format and read.

(b) What are the key words to handle exceptions in Python?

> "try" and "except" are the python keywords that enable us to handle exceptions. A block of code can be placed in try, and exception can be run if their was an error running the code within try. The exception can also be specified so different exceptions occur depending on the error.

(c) How do you define a function in Python?

> A function in python is similar to functions from other languages. "def" is used to indicate a function with the function name coming afterwards, and parameters in "()". You do not have to define the return or parameter variable type.

(d) True or False - when calling a specific function, the number of arguments is fixed.

> False, Python functions can be specified to have arbitrary arguments when you don't know how many values you want to pass into the function making the number of arguments non fixed.

(e) What are two methods that can be used with list objects, and what do those methods do?

> Two methods available to use with list objects are "reverse" and "pop". The "reverse" method will reverse the order of objects within the list. The "pop" method removes values from a list at a specific index, and returns what that value was.

(f) Carefully describe all the differences between the example state machine implemented with if statements, and the example state machine implemented with function pointers. How do pointers make it easier to add new states?

> The first example state machine gives each state an integer value, and is reliant on a large if statement within main to match the state integer value and the state function. The second example uses pointers to select which states code is being run. This second approach is much cleaner because you don't need to modify a large if statement to add a state, and it is much easier to give each state a name. Instead of "return 0" you can do "return runState" which is much clearer to understand.

Code for Exercise 1:

```
#   Eric Sobczak
#   8/26/2022
#   SEED Lab Python Assignment 1, Excercise 1

# This program reads throught the values in " datafile.txt" and outputs
# information on values within. The program lists both the largest and
# smallest values in the datafile. The program also finds all the
# locations of a particular values and gives you their index within the
# list. It is currently set to look for 38 in "indexToFind" variable.
# Then, repeated values are listed. The vzlues that are listed can be
# adjusted with the "repeatedBreak" variable. Lastly, the lists are sorted
# and also separated to just even. All these tasks are completed using
# Numpy. The datafile.txt can be changed to a different filetype by changing
# the "datafile" variable.

##########   IMPORT AND CONSTANTS   ##########
# Import the numpy package as np
import numpy as np

#Constants
#Name of file containing integars
datafile = "datafile.txt"
#What value to find the index of within the array
indexToFind = 38
#Determine mininum number of repeats to be considered relevant
repeatedBreak = 3

##########   FUNCTIONS   ##########
#Code for finding Max
def findMax(np_array):
    if np_array.size > 0:
        return np.amax(np_array)
    else:
        Exception("Cant find max of empty array")

#Code for finding Min
def findMin(np_array):
    if np_array.size > 0:
        return np.amin(np_array)
    else:
        Exception("Cant find min of empty array")

#Code for finding index of a value
```

```
def findValue(np_array, value):
    tempLocal =  np.where(np_array == value)[0]
    #Return string with info, but format depending on count
    if tempLocal.size > 1:
        return ("can be found at the indexes " + str(tempLocal))
    elif tempLocal.size == 1:
        return ("can be found at the index " + str(tempLocal[0]))
    else:
        return ("cannot be found")

#Repeated numbers view
def repeatedValues(np_array):
    #Sort original list so preview looks nice
    np_sorted = np.sort(np_array)
    #Use Numpy Unique to find which value is most common
    np_unique = np.unique(np_sorted, return_counts=True)
    np_unique_values = np_unique[0]
    np_unique_counts = np_unique[1]
    #Calculate sort permutation on unique counts
    np_counts_perm = np_unique_counts.argsort()
    #Apply sort prmutation and flip
    np_values_sorted = np.flip(np_unique_values[np_counts_perm])
    np_counts_sorted = np.flip(np_unique_counts[np_counts_perm])
    #Return two new arrays for printing
    return([np_values_sorted, np_counts_sorted])

#Sorted List
def sortList(np_array):
    return np.sort(np_array)

#All even numbers, in order,
def sortEven(np_array):
    #Use list comprehension to limit to only even values.
    np_even_values =[value for value in np_array if value %2 == 0]
    return np.sort(np_even_values)

##########   INITIALIZATION   ##########
#Open the datafile containg integars
try:
    with open(datafile,'r') as f:
        integarList = eval(f.read())
    #Convert the list into a Numpy List
    np_integarList = np.array(integarList)
except:
    #Raise exception if file is bad
```

```
    raise Exception("Invalid file!")

#Print Max, Min, and Index
print("The Maximum value within the list of integars is",
findMax(np_integarList))
print("The Mininum value within the list of integars is",
findMin(np_integarList))
print("The value", indexToFind, findValue(np_integarList, indexToFind))

#Print repeated values
np_repeated_info = repeatedValues(np_integarList)
if (np_repeated_info[1][0] > 1):
    print("\nThis list contains repeated value and their frequency")
    print("Value \t|  # of repeats")
    for index, x in enumerate(np_repeated_info[1]):
        #Only prints values above the threshold
        if x >= repeatedBreak:
            print(" "  + str(np_repeated_info[0][index]) + "\t|   " + str(x))
else:
    print("\nThis list contains no reaptead values")

#Print sorted array:
print("\nList of integars sorted using Numpy:", end =" ")
np_sorted_list = sortList(np_integarList)
for index, x in enumerate(np_sorted_list):
    #Create a new line after 10 values
    if index % 10 == 0:
        print()
    print(x, end ="\t ")

#Print even numbers in order
print("\n\nSorted even numbers from list using list comprehension", end =" ")
np_sorted_even = sortEven(np_integarList)
for index, x in enumerate(np_sorted_even):
    #Create a new line after 10 values
    if index % 10 == 0:
        print()
    print(x, end ="\t ")

##########   SOURCES USED   ##########
#Numpy Max, Min, Size, Where, Flip, Sort, Argsort:
# https://numpy.org/doc/stable/reference/generated/numpy.amax.html
# https://numpy.org/doc/stable/reference/generated/numpy.amin.html
# https://numpy.org/doc/stable/reference/generated/numpy.ndarray.size.html
# https://numpy.org/doc/stable/reference/generated/numpy.where.html
```

```
# https://numpy.org/doc/stable/reference/generated/numpy.flip.html
# https://numpy.org/doc/stable/reference/generated/numpy.sort.html
# https://numpy.org/doc/stable/reference/generated/numpy.argsort.html
#
#How to calculate and apply sort permutation
# https://exchangetuts.com/how-can-i-zip-sort-parallel-numpy-arrays-
1639552927939566
#
#Tabs in python
# https://www.delftstack.com/howto/python/python-print-tab/
#
#Print without newline
# https://www.geeksforgeeks.org/print-without-newline-python/
#
#Exception handeling tips
# https://www.w3schools.com/python/python_try_except.asp
```

Code for Exercise 2:

```
#   Eric Sobczak
#   8/28/2022
#   SEED Lab Python Assignment 1, Excercise 2

# This program uses a Finite State Machine to detect the sequence
# "abcd" within a user entered string. When the program runs, the
# user is asked to input a string and then hit enter. Then the
# code goes letter by letter through the entered string and passes
# them to a state machine. If the FSM machine detects "abcd" it will
# list out its occurrences, otherwise it will say none were found.

##########   FSM STATES   ##########
def state0(nextLetter):
    #If A was found go to A
    if nextLetter == "a":
        return stateA
    #Stay at zero becuase irrelevent next letter
    else:
        return state0

def stateA(nextLetter):
    #A was found, go back to start
    if nextLetter == "a":
        return stateA
    #If B was found go to B
    elif nextLetter == "b":
        return stateB
    #Go back to zero becuase irrelevent next letter
    else:
        return state0

def stateB(nextLetter):
    #A was found, go back to start
    if nextLetter == "a":
        return stateA
    #If C was found go to C
    elif nextLetter == "c":
        return stateC
    #Go back to zero becuase irrelevent next letter
    else:
        return state0

def stateC(nextLetter):
```

```
    #A was found, go back to start
    if nextLetter == "a":
        return stateA
    #If D was found go to D
    elif nextLetter == "d":
        return stateD
    #Go back to zero becuase irrelevent next letter
    else:
        return state0

def stateD(nextLetter):
    #A was found, go back to start
    if nextLetter == "a":
        return stateA
    #Return to 0 becuase no string
    else:
        return state0

#Dictionary containg all states
state_dictionary = {
    state0 : "Searching",
    stateA : "A Found",
    stateB : "B Found",
    stateC : "C Found",
    stateD : "D Found"
    }

##########   INITIALIZATION   ##########
state = state0 #initial state
foundLocations = [] #Holder for all found locations

print("Enter a string of text, and the FSM will look for 'abcd'")
#Collect input from user
inputString = input("input: ")

#Loop through all characters in string
for index, x in enumerate(inputString):
    new_state = state(x)
    state = new_state
    #If the FSM has reached state D, the string was found
    if (state == stateD):
        #Markdown the index of where the string occured
        foundLocations.append((index-3))

#Loop through found locations and print
```

```python
if (len(foundLocations) == 0):
    #If none where found, print none
    print("No 'abcd' was found")
else:
    #Print found with index of 'abcd'
    for i in foundLocations:
        print("'abcd' found at the index", i)

##########   SOURCES USED   ##########
#I forgot how to iterate through a string in python,
#forogt for loop doesnt require anything additional
# https://www.geeksforgeeks.org/iterate-over-characters-of-a-string-in-python/
```