Lab 01 - Getting Started

Name: Kartikeya Sharma

Class: CSCI 349 - Intro to Data Mining

Semester: 2021SP

Instructor: Brian King

In [1]:
```
import sys
print(sys.version)
```

```
3.8.8 (default, Feb 24 2021, 13:46:16)
[Clang 10.0.0 ]
```

1. First, fill out your README.md file with some basic biographical info. Your main Gitlab page must state who you are, the course info, your skills, and a sentence or two about what you are hoping to achieve after graduation. Commit and push your README.md file

README.md committed and pushed to gitlab.

2. [M] Did you read the syllabus? All of it? Do you agree to abide by the cheating rules? Write a sentence clearly indicating your commitment to not cheating.

Yes, I read all of the syllabus, and I agree to abide by the cheating rules; I strive to be a student of integrity, following the spirit of the Academic Responsibility principles and Bucknell University Honor Code highlighted in this course's stin the highest regard.

3. [M] What are you hoping to get out of this course?

In this course, I hope to use the integrative experience of absorbing the class materials, working through assignments, and applying the web of knowledge acquired to broader-scope projects. Within this integrative experience, I intend to acquire, refine, and apply skills in data science and machine learning to understand patterns within data, including time-series/sequential data.

With regards to applying these skills to a broader-scope project beyond this course, I have been working under the guidance of Professor King on designing a software toolkit that analyzes, and subsequently provides a visualization of, eye tracking data from experiments presenting a visual stimulus with an assortment of spatially-separated entities/objects to children with and without autism spectrum disorder. Prior to presenting a visualization of the eye tracking data, the data should utilize 'preprocessing and cleaning,' which is bolded as one of the fundamental topics of this course within the course syllabus, to then display a more clear visual. Afterwards, understanding trajectories in eye movement, sequence of objects focused upon, etc. requires a deep understanding of trajectory pattern mining, which the course material focused around pattern mining will expose me to important fundamentals for further learning about trajectory pattern mining, a more specialized pattern mining endeavor. Furthermore, the project involves defining areas of interest on the image stimulus exposed to the participants of the experiments through algorithm(s) within program(s) integrating clustering and machine learning. Due to the course covering clustering as one of the final stages of the course outlined in the syllabus, it will lend itself to the machine learning and clustering portion of this software toolkit and associated algorithm(s) design, which is planned to be wrapped into my capstone thesis at Bucknell.

In conclusion, this advanced computer science elective in the field of data mining course is meaningful in providing me with a sphere of data science background and associated toolkits in the field of data mining. This acquired knowledge, then, can guide me further in my learning process and discovery within my gradually-found, specialized interest in the intersection of pattern mining and machine learning.

4. [P] Print the Python version (available in sys package)

In [2]:
```
import sys
print(sys.version)
```

```
3.8.8 (default, Feb 24 2021, 13:46:16)
[Clang 10.0.0 ]
```

5. [P] Create a Python list of 10000 random integers in the range 1 to 100, using the random package. Name the list x_list.

In [3]:

```
# importing randint function from random package without
# unnecessarily importing every (public) function within
# the random package
from random import randint


# parameters created to increase readability and adaptability
min_bound = 1
max_bound = 100
num_int = 10000

x_list = [randint(min_bound, max_bound) \
          for i in range(num_int)]
```

6. [P] What is the minimum value of x_list? What is the max value? What is the mode?

In [4]:

```
from statistics import mode

print("The minimum value of x_list is " + \
      str(min(x_list)) + ".")
print("The maximum value of x_list is " + \
      str(max(x_list)) + ".")
print("The mode of x_list is " + \
      str(mode(x_list)) + ".")
```

```
The minimum value of x_list is 1.
The maximum value of x_list is 100.
The mode of x_list is 37.
```

7. [P] Write a function to take a list of numbers as a parameter, and return the average of the list. Then, use your function to report the average value of x_list, printed as a float with 2 places of precision.

In [5]:
```
def get_list_average(list_nums):
    """
    Returns the average value of x_list as a float

    Parameters:
    list_nums (list): list of numbers to retrieve
                      the average of
    """
    sum = 0
    for num in list_nums:
       sum += num
    average = sum/len(list_nums)
    return average

def print_float(float_num, num_places_precision=2):
    """
    Prints the inputted float with a precision of
    the number of decimal places provided (default: 2
    decimal places)

    Parameters:
    float_num (list): float number to be printed
    num_places_precision (int): number of decimal places
                                of precision to which
                                the inputted average value
                                will be printed
                                (default: 2)

    """
    format_prefix = "{:." + str(num_places_precision) + "f}"
    formatted_float_num = format_prefix.format(float_num)
    print(formatted_float_num)

x_list_average = get_list_average(x_list)
print_float(x_list_average)
```

```
50.72
```

8. [P] Create a list called x_hist that represents a histogram, i.e. a distribution of the numerical data, of x_list. Each entry in x_hist should contain the range of data in widths of 10. So, x_hist[0] represents the frequency of numbers between 1-10, x_hist[1] is the frequency of numbers between 11-20, and so on. Be sure to print x_hist at the end.

In [6]:
```
def get_hist_list_width_of_10(input_list):
    """
    Returns a list representing a histogram of bucket width 10

    Parameters:
    input_list (list): list of positive integers containing
    """
    hist_list = [0 for i in range(10)]
    for value in input_list:
        hist_list[int((value-1)/10)] += 1
    return hist_list

print(get_hist_list_width_of_10(x_list))
```

```
[989, 1002, 974, 978, 987, 1024, 1029, 1027, 1003, 987]
```

9. [M] What is numpy? What are its strengths? Does it have any weaknesses?

numpy is a widely-adopted package in Python that contains a collection of methods and functions for "scientific computing," particularly for computations that are mathematical or statistical. It emphasized object-orientedness through its central usage of multidimensional array objects.

The primary advantage of using this package is its providing a large range of data science methods and functions for those individuals who are focused in other disciplines. Another advantage is that it further increases the already relatively high level of readability in Python. It does this by decreasing the number of lines of codes that programmers need to write due to its availability of a suite of methods and functions to manage the sets of data, for instance, contained in arrays. Also, numpy fixes the size of its arrays (changing the size of a numpy array creates a new array in memory with the new size, copying the items from the old array into the new one, and deleting the old array from memory), increasing the predictability of the behavior of manipulations of the underlying numpy array. Similarly, numpy fixes the data type of items in a particular numpy array, in turn, fixing the the memory size assigned to each item in the array and, consequently, further increasing predictability. Finally, numpy contains the advantages of Python and C, adopting the simplicity and relatively high readability of Python while also adopting the efficient memory management and execution of mathematical operations.

On the other hand, numpy has its own drawbacks. For one, because numpy arrays and objects are stored in contiguous blocks of memory, expanding or shrinking the size of an array requires allocating a new space in memory with respect to the new size of the array, copying the elements from the old array to the newly allocated array, and deallocating the old array, which is, in turn, overhead. In addition, there is ambiguity in dealing with undefined items in a numpy array, where NaN items assigned by numpy to undefined items are not the same as null; NaN items are not all equal because they are produced through different means (hence, they are represented by different float values), but they are given the same ID by Python, so one can verify a float as NaN by its id. Other platforms do not necessarily use NaN (and instead resort to null, for instance, for undefined items); hence, compatibility across platforms when undefined items in an array are involved can be a potential issue. Finally, for someone who is only interested in doing simple array operations, learning and implementing numpy could be more overhead than needed for a light Python user.

Sources

1. https://numpy.org/doc/stable/user/whatisnumpy.html
2. https://www.educba.com/introduction-to-numpy/
3. https://towardsdatascience.com/navigating-the-hell-of-nans-in-python-71b12558895b

10. [P] Import the numpy package as np and print the numpy version

In [7]:
```
import numpy as np
print(np.__version__)
```

```
1.19.2
```

11. [M] What is the primary object type in numpy? Can it store data of different types? Discuss.

The primary object in a numpy array, which is an n-dimensional array that stores arrays of related data types of the same size and same type. Numpy arrays in general can store collections of data that are of various types, such as floating point values, integer values, and other objects. Each numpy array (numpy.ndarray), however, has its own instance of numpy.dtype, which is a class that defines that type of the items in that particular numpy array, restricting each numpy array to holding items of one data type.

Sources

1. https://numpy.org/doc/stable/user/whatisnumpy.html
2. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html

12. [M] Discuss the types of data available in numpy. You need not list every type. Just generalize, and discuss how the type system is different than the built-in types int and float in Python.

The types of data available in numpy, as obtained from W3Schools, include numerical values, a variety of types of strings[1], a variety of types of objects, and those representing a "fixed chunk of memory."[2] The importance of having integer and float values represented by numpy's int and float types rather than by the built-in ones in Python is that the numpy types allow for the numpy array to ensure that each element in the array is of the same type.[3] even distinguishing shorter ints from longer ints like int8 vs. int16 (8-bit vs. 16-bit integers, respectively).[3] Also, NumPy scalars, including NumPy ints and NumPy floats, have "many of the same methods arrays do."[3] Moreover, the numpy data types are used to represent numerical python variables in a way that can be inserted into a numpy array, after which numerical computations can be done with relatively higher efficiency (than would be done in python) leveraging the underlying C low-level code that it interfaces with for such numerical computations.[3]

Sources

1. https://numpy.org/doc/stable/reference/arrays.datetime.html
2. https://www.w3schools.com/python/numpy_data_types.asp
3. https://numpy.org/doc/stable/user/basics.types.html

13. [P] Create a numpy array from x_list. Reassign it as x_list. Show the contents.

In [8]:
```
x_list = np.array(x_list)
print(x_list)
```

```
[77 41 95 ... 45 27 60]
```

14. [P] Redo the previous exercise, but set the data type of each value to 'float32'

In [9]:
```
x_list = np.array(x_list, dtype='float32')
print(x_list)
```

```
[77. 41. 95. ... 45. 27. 60.]
```

15. [P] Create a length 10 integer array filled with zeros

In [10]:
```
np.zeros(10, dtype='int')
```

Out[10]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

16. [P] Create a float array of 3 rows and 4 columns, all initialized to one.

In [11]:

```
np.ones((3,4), dtype='float')
```

Out[11]:

```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

17. [P] Create an array of 20 values, evenly spaced between 1 and 3 (HINT – look at np.linspace)

In [12]:

```
# between is assumed to be including both 1 and 3
np.linspace(1,3,num=20)
```

Out[12]:

```
array([1.        , 1.10526316, 1.21052632, 1.31578947, 1.42105263,
       1.52631579, 1.63157895, 1.73684211, 1.84210526, 1.94736842,
       2.05263158, 2.15789474, 2.26315789, 2.36842105, 2.47368421,
       2.57894737, 2.68421053, 2.78947368, 2.89473684, 3.        ])
```

18. [P] Set the numpy random seed to the value 12345, then create a 10 x 5 array of random integers on the interval [10, 20)

In [13]:

```
np.random.seed(seed=12345)
np.random.randint(10,20,(10,5))
```

Out[13]:

```
array([[12, 15, 11, 14, 19],
       [15, 12, 11, 16, 11],
       [19, 17, 16, 10, 12],
       [19, 11, 12, 16, 17],
       [17, 17, 18, 17, 11],
       [17, 14, 10, 13, 15],
       [17, 13, 11, 15, 12],
       [15, 13, 18, 15, 12],
       [15, 13, 10, 16, 18],
       [10, 15, 16, 18, 19]])
```