CSE 4554
Machine Learning Lab

Experiment No: 1

Name of the experiment: Introduction to Python for Machine Learning

Hasan Mahmud, Ph.D.
Associate Professor, Department of CSE, IUT
Md. Shihab Shahriar
Lecturer, Department of CSE, IUT

August 24, 2023

Contents

1 Objectives

- To learn the basics of Python Programming which mainly focuses in ML
- Introduction of the Python Environments, Editors
- Introducing the Python Library Numpy
- Applying Numpy in different tasks of manipulating Array
- Using Numpy to store the IRIS dataset

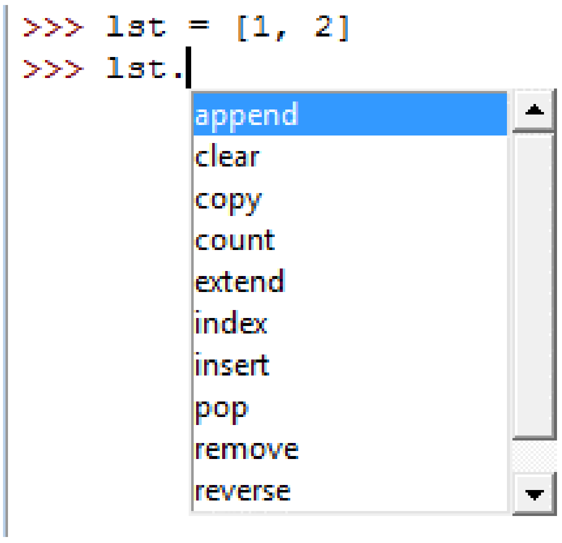
2 Problem Discussion

2.1 Introduction to Python

Get to know about basic python:

I. Python List and tuple, function calling, if else scope, loops and many more `https://www.programiz.com/python-programming/list`

II. Variable name followed by dot then press Tab to see the available option



```
>>> lst = [1, 2]
>>> lst.
```

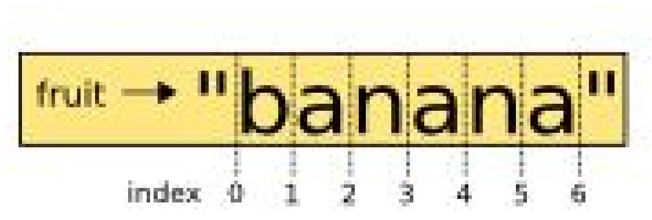
III. Negative indexing, which counts backward from the end of the sequence. The expression seq[-1] yields the last element, seq[-2] yields the second to last, and so on.

```
classmates=["Alejandro", "Ed", "Kathryn", "Presila", "Sean", "Pet"]
classmates[-5]
# 'Ed'
word = "Alphabet"
word[-3]
# 'b'
```

IV. Slicing. Like with indexing, we use square brackets ([ ]) as the slice operator, but instead of one integer value inside we have two, separated by a colon (:)

```
singers="Peter, Paul"
singers[0:5]
# 'Peter'
classmates=["Alejandro", "Ed", "Kathryn", "Presila", "Sean", "Pet"]
classmates[2:4]
# ['Kathryn', 'Presila']
```

The operator [n:m] returns the part of the sequence from the n'th element to the m'th element, including the first but excluding the last.



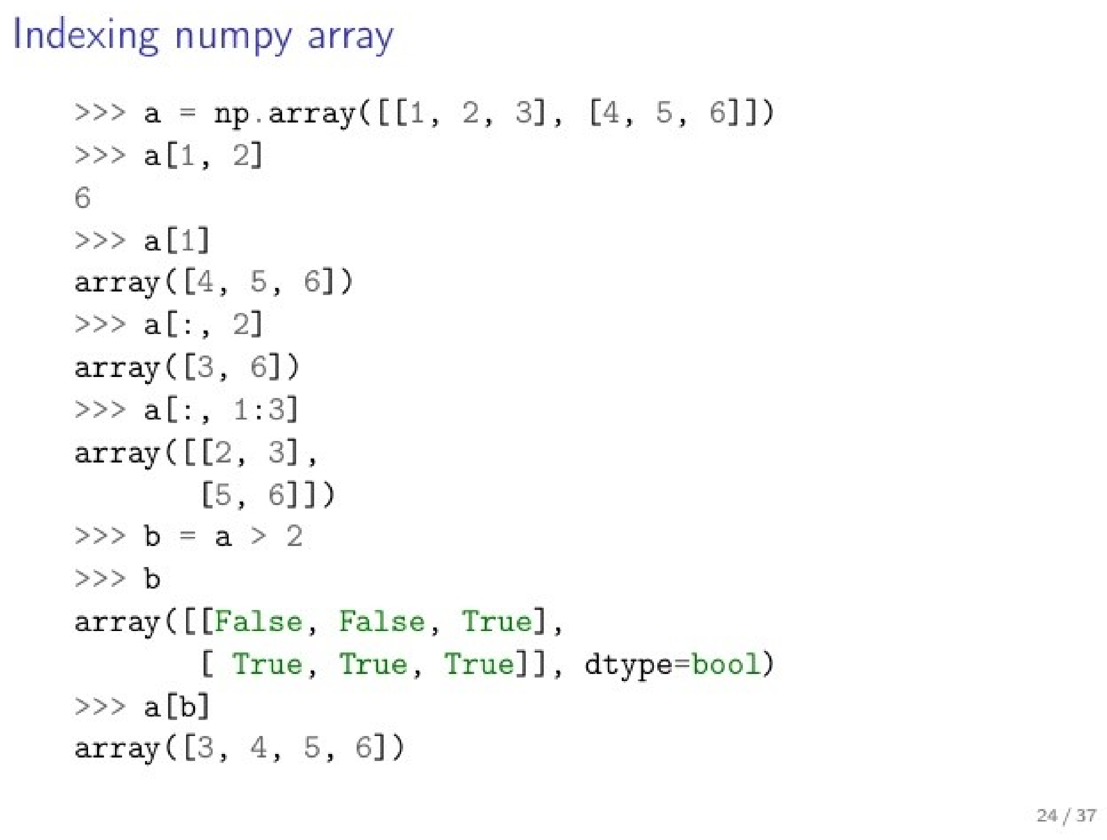
If you omit the first index (before the colon), the slice starts at the beginning of the string. If you omit the second index, the slice goes to the end of the string. Thus:

```
fruit="banana"
fruit[:3]
# 'ban'
fruit[3:]
# 'ana'
```

What do you think s[:] means? It means all the elements
Negative indexes are also allowed, so

```
fruit[-2:]
# 'na'
classmates[:-2]
# ['Alejandro','Ed','Kathryn','Presila']
```

V. To handle multidimensional array Numpy array library is used, however library need to import first then using the "as" command you can create an instance for the librar "Import numpy as np"
`https://jakevdp.github.io/PythonDataScienceHandbook/02.02-the-basics-of-numpy-array`
`html`



Indexing numpy array

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> a[1, 2]
6
>>> a[1]
array([4, 5, 6])
>>> a[:, 2]
array([3, 6])
>>> a[:, 1:3]
array([[2, 3],
       [5, 6]])
>>> b = a > 2
>>> b
array([[False, False, True],
       [ True, True, True]], dtype=bool)
>>> a[b]
array([3, 4, 5, 6])
```

24 / 37

2.2 Introduction to Numpy

NumPy is one of the fundamental packages for scientific computing in Python. It contains functionality for multidimensional arrays, high-level mathematical functions such as linear algebra operations and the Fourier transform, and pseudorandom number generators. In scikit-learn, the NumPy array is the fundamental data structure. scikit-learn takes in data in the form of NumPy arrays. Any data you're using will have to be converted to a NumPy array. The core functionality of NumPy is the ndarray class, a multidimensional (n-dimensional) array. All elements of the array must be of the same type. NumPy arrays form the core of nearly the entire ecosystem of data science tools in Python. Let us practice some example Numpy codes.

I. **Python Array list**
   a = [1,2,3,4,5,6,7,8,9]
   Numpy array list
   A = np.array([1,2,3,4,5,6,7,8,9])

   A = np.arange(0,10,2)
   A is array([0, 2, 4, 6, 8])

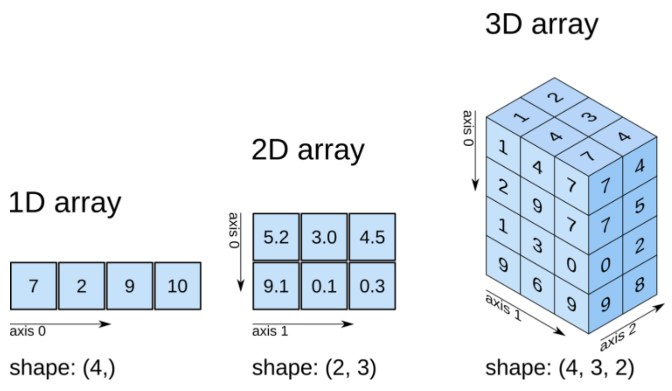
II. Shape is an attribute for np array. When a default array, say for example A is called with shape, here is how it looks.
   A = [1, 2, 3, 4, 5, 6, 7, 8, 9]
   A.shape
   –> (9,)

III. This is a rank 1 matrix(array), where it just has 9 elements in a row. reshape() that changes the dimensions of your original matrix into your desired dimension.
   A.reshape(1,9)

```
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
```

IV. Two square brackets in the beginning indicate that. [[1, 2, 3, 4, 5, 6, 7, 8, 9]] is a potentially multi-dim matrix as opposed to [1, 2, 3, 4, 5, 6, 7, 8, 9].

V. To understand array Shape write commandvariablename.shape or to change dimension write variablename.reshape(x value,y value,z value)



VI. Numpy array attributes

```python
import numpy as np
np.random.seed(0)

x1 = np.random.randint(10, size=6) # one dimensional array
x2 = np.random.randint(10, size=(3,4)) # two dimensional array
# similarly, you can create a three dimensional array
```

Each array has attributes ndim (the number of dimensions), shape (the size of each dimension), and size (the total size of the array):

```python
print("x3 ndim:", x3.ndim)
print("x3 shape:", x3.shape)
print("x3 size:", x3.size)
```

VII. One dimensional sub arrays

```python
x=np.arange(10)
x[::2] # every other element
array([0,2,4,6,8])
x[1::2] # every other element starting at index 1
x[::-1] # all elements reversed
```

VIII. Multi-dimensional sub arrays

```python
x = np.array([[12, 5, 2, 4],
              [7, 6, 8, 8],
              [1, 6, 7, 7]])
x2[:2, :3] # two rows, three columns
print(x2[0]) # equivalent to x2[0,:]
```

IX. Creating arrays: np.zeros((n,m)) returns an n x m matrix that contains zeros. It's as simple as that

```python
np.zeros((4,3))
# array([[12, 5, 2, 4],
        [7, 6, 8, 8],
        [1, 6, 7, 7]])
```

X. Concatenation of arrays

```python
x=np.array([1,2,3]
y=np.array([3,2,1]
np.concatenate([x,y])
```

XI. To read csv files using numpy
1) **white_winesn̄p.genfromtxt("winequality-white.csv", delimiter"̄,", skip_header1̄)**

If the data columns are separated with comma use following command, also this command will replace the non float point values with -999
2) **data = np.genfromtxt(path, delimiter=',', skip_header=1, filling_values=-999, dtype='float')**

You can just do the following if your dataset does have any Text
3) **data2 = np.genfromtxt(path, delimiter=',', skip_header=1, dtype=None)**

3 Tasks

- **Task 01:**
  Create a one dimensional random numpy array of size 10.
- **Task 02:**
  Create a two dimensional numpy array of size 7x6 and reshape it to 3x14 array.
- **Task 03:**
  Create an array and extract all the odd numbers from it. [array([0,1,2,3,4,5,6,7,8,9])].
- **Task 04:**
  Replace all the odd numbers in the previous array with -1.
  ==> array([0, -1, 2, -1, 4, -1, 6, -1, 8, -1])
- **Task 05:**
  Replace all the odd numbers in the previous array with -1 without affecting the original array.
  arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
  out
  #> array([0, -1, 2, -1, 4, -1, 6, -1, 8, -1])

arr
#> array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

- **Task 06:**
  Get the positions where elements of a and b match
  a = np.array([1,2,3,2,3,4,3,4,5,6])
  b = np.array([7,2,10,2,7,4,9,4,9,8])

- **Task 07:**
  Get all the items between 5 and 10 from a.
  a = np.array([2, 6, 1, 9, 10, 3, 27])

- **Task 08:**
  Create a vector of size 10 with zeros. Then create a vector of size 10 with ones. Change their dimension and after that merge the the two vector vertically and horizontally using "concatenate".

- **Task 09:**
  Create an ndarray x of shape= (5, 4) with random numbers – Each of the 5 rows represents a sample with 4 features

- **Task 10:**
  2. Create a flat ndarray y of shape=(5,), whose elements are 0 and 1 – Each element is the class label of the corresponding sample in x [Hint: use np.random.rand(...) and np.random.randint(...)]
  import numpy as np
  n_samples = 5
  n_features = 4
  x = np.random.rand(n_samples, n_features)
  y = np.random.randint(0, 2, n_samples)
  print 'x =', x
  print 'y =', y

- **Task 11:**
  Define a function extract_subset(x, y, y0) that takes as input: – the feature matrix x, and the labels y from the previous exercise – a target class y0 (i.e., either y0=0 or y0=1) and returns a feature matrix containing only samples belonging to y0

- **Task 12:**
  Import a dataset (IRIS dataset) by keeping the text and numbers intact.

- **Task 13:**
  Find the mean, median and standard deviation of the 1st column of the given dataset. Give statistical summary of the dataset as many as possible.

4 Submission

Submit Your Google Colab notebook in the classroom with the following naming format <Student_id>_Lab<Lab_id>.