

Graph Mining - CSF426

Lab Session - 1

Date - 26/08/2024

Marks - Non-evaluative

Instructor IC: Vinti Agarwal

✓ Introduction to Python

Python is object-oriented high-level language, designed by Guido van Rossum and first released in 1991. Python boasts an extensive standard library and a large number of third-party libraries, making it the preferred language for many developers in data science and machine learning.

Check version of python

```
!python --version
```

```
Python 3.10.12
```

```
!jupyter --version
```

```
Selected Jupyter core packages...
IPython          : 7.34.0
ipykernel       : 5.5.6
ipywidgets      : 7.7.1
jupyter_client  : 6.1.12
jupyter_core    : 5.7.2
jupyter_server  : 1.24.0
jupyterlab      : not installed
nbclient        : 0.10.0
nbconvert       : 6.5.4
nbformat        : 5.10.4
notebook        : 6.5.5
qtconsole       : not installed
traitlets       : 5.7.1
```

Install a package

```
!pip install numpy
```

```
#NOTE: Use !pip to install a package in colab notebook or jupyter notebook, whereas use pip to install package in command prompt
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
```

```
# After installing - import the package
```

```
# Syntax - import <package-name> or import <package_name> as <alias>
```

```
import numpy as np
```

✓ Data types:

Built-in Data Types

- Python has the following data types built-in by default:

1. **Text Type:** str

2. **Numeric Types:** int, float, complex

3. **Sequence Types:** list, tuple, range

4. **Mapping Type:** dict

5. **Set Types:** set, frozenset

6. **Boolean Type:** bool

7. **Binary Types:** bytes, bytearray, memoryview

NOTE: Use type() function to get the data type of any variable

Q1: What is the difference between lists, arrays, and sets in Python, and when should we use each of them?

Answer:

These are three distinct types of data structures that can hold sequences of data, each with their unique properties.

Lists are denoted by square brackets [], sets by curly braces {}, and arrays by parentheses ().

List: Lists are used to store multiple items in a single variable. List items are ordered, changeable (support adding and removing of items), and allow duplicate values. List items are indexed, the first item has index [0], the second item has index [1], so on. One of its benefits is that it allows the use of many data types in the same lists as it can store string, integers, floats of any other derived objects. One of its cons that lists are very slow if it will be used in numerical computation.

Array: Arrays are tailored to hold items of the same data type only (integer only, float only etc.), making them highly efficient in terms of speed and memory use, especially useful in numerical computations. *NumPy*, a popular library known for its array operations, significantly enhances Python's capabilities in handling numerical data.

Set: This Python built-in data type is designed to hold unique elements, automatically removing any duplicates. Sets support a variety of operations, such as unions, differences, and intersections.

```
# Create List of fruits
fruits = ["Apple", "Mango", "Banana", "Grapes", "Pear", "Apple"]
print(fruits)
print(type(fruits))
```

```
['Apple', 'Mango', 'Banana', 'Grapes', 'Pear', 'Apple']
<class 'list'>
```

```
# create an array of 10 numbers
arr = np.array([1,2,2,4,4,4,7,3,9,10])
print(arr)
print(type(arr))
```

```
[ 1  2  2  4  4  4  7  3  9 10]
<class 'numpy.ndarray'>
```

```
# create a set of 10 numbers
set = {1,2,2,4,4,4,7,3,9,10}
print(set)
print(type(set))
```

```
{1, 2, 3, 4, 7, 9, 10}
<class 'set'>
```

Q2. What is the difference between a Set and Dictionary?

The set is an unordered collection of data types that is iterable, mutable and has no duplicate elements.

A dictionary in Python is an ordered collection of data values, used to store data values in [key:values](#) pair.

```
dict = {"name": "John", "age": 25}
print(dict)
print(dict['name']) # access the value corresponding to "name" key
print(type(dict))
```

```
{'name': 'John', 'age': 25}
John
<class 'dict'>
```

Q3. What is the difference between / and // in Python?

```
## write your code here
```

Q4. What is the difference between append, insert and extend operations on a list? Use these operations on "fruits" list and print results.

```
# write your code
```

Q5. What is list comprehension? How is it used for accessing elements of a list?

Answer:

List comprehension allows a short method to loop through the elements of a list.

Suppose we want to create a new list from "fruits" list for those fruits that have "an" in its name.

Without list comprehension you will have to write a for statement with a conditional test inside it:

```
new_fruits = []
for fruit in fruits:
    if "an" in fruit:
        new_fruits.append(fruit)
print(new_fruits)
```

```
↳ ['Mango', 'Banana', 'Orange']
```

With list comprehension you can do all that with only one line of code:

```
newlist = [x for x in fruits if "an" in x]
print(newlist)
```

```
↳ ['Mango', 'Banana', 'Orange']
```

Q6. What are generators in Python? how they are useful in saving memory?

Ans: Generator functions are a special kind of function that return a lazy iterator. These are objects that you can loop over like a list. However, unlike lists, lazy iterators do not store their contents in memory. Generators are useful when we want to produce a large sequence of values, but we don't want to store all of them in memory at once.

Suppose we want to read a very large file, but don't have enough memory space to accommodate the file contents at once. This is where generators come in handy.

```
# Common way of reading a file
def csv_reader(file_name):
    file = open(file_name)
    result = file.read().split("\n")
    return result
```

```
csv_gen = csv_reader("some_csv.txt")
row_count = 0
for row in csv_gen:
    row_count += 1
```

```
print(f"Row count is {row_count}")
```

```
## To populate this list, csv_reader() opens a file and loads its contents into csv_gen. Then, the program iterates over the list and iterates over the list.
## However, this approach will face difficulty when reading a large file. In such cases, file.read().split() loads everything into memory.
```

```
# New csv_reader function
def csv_reader(file_name):
    for row in open(file_name, "r"):
        yield row
```

```
## This version of csv_reader function opens a file, loops through each line, and yields each row, instead of returning it.
## yield row statement returns a single row from the file each time the generator is iterated over.
```

Q7 What is swapcase function in Python? Use swapcase function on the string "Birla Institute"

```
string = "Birla Institute"
# print swapcase results
```

Q8. How are classes and objects created in Python?

```
# Classes and Objects
class Dog:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def bark(self):
        print("Woof!")

my_dog = Dog("Buddy", "Golden Retriever")
print(my_dog.name) # Accessing object properties
print(my_dog.breed) # Accessing object properties
my_dog.bark() # calling class function
```

```
↳ Buddy
Golden Retriever
Woof!
```

***NOTE:** *init* is a method or constructor in Python. This method is automatically called to allocate memory when a new object/ instance of a class is created. All classes have the *init* method.

Q9 What is the function of "self"?

write your answer here

Q10. How does break, continue and pass work?

Break: Allows loop termination when some condition is met and the control is transferred to the next statement.

Continue: Allows skipping some part of a loop when some specific condition is met and the control is transferred to the beginning of the loop.

Pass: Used when you need some block of code syntactically, but you want to skip its execution. This is basically a null operation. Nothing happens when this is executed.

```
for letter in 'Python':
    if letter == 'h':
        break
    print('Current Letter :', letter)
```

```
↩ Current Letter : P
  Current Letter : y
  Current Letter : t
```

```
for letter in 'Python':
    if letter == 'h':
        continue
    print('Current Letter :', letter)
```

```
↩ Current Letter : P
  Current Letter : y
  Current Letter : t
  Current Letter : o
  Current Letter : n
```

```
for letter in 'Python':
    if letter == 'h':
        pass
    print('This is pass block') # do nothing
    print('Current Letter :', letter)
```

```
↩ Current Letter : P
  Current Letter : y
  Current Letter : t
  This is pass block
  Current Letter : h
  Current Letter : o
  Current Letter : n
```

Q11. How to implement case sensitive sorting in the following list?

```
veggies = ["spinach","Onion","potato","Raddish"]
```

```
veggies = ["spinach","Onion","potato","Raddish"]
veggies.sort(_____) ##### fill the blank
print(veggies)
```

```
↩ ['Onion', 'potato', 'Raddish', 'spinach']
```

sort() function can be customized also to implement any operation. For example: Sort the list based on how close the number is to 500:

```
def myfun(n):
    return abs(n - 500)
```

```
List = [1000, 503, 650, 812, 23]
List.sort(_____) ##### fill the blank
print(List)
```

```
↩ [503, 650, 812, 23, 1000]
```

Q12. Define tuples and lists in Python. When do we prefer tuples over lists?

Answer: Lists are mutable, are better for performing operations, such as insertion and deletion. Whereas tuples are immutable and ordered. Tuples are sequences, just like lists. The differences between tuples and lists is that tuples cannot be modified unlike lists.

A tuple takes up less memory space than a list, so whenever an array of objects is known to be immutable, it is recommended to use tuples instead of lists.

Q13. How is split() function used to divide a string?

```
string = "Hi. My name is John. Nice to meet you."  
## write code to split the above string based on full stop " . " token
```

Q14. How are local and global variables used in Python?

Answer:

When a variable is defined outside of a function, it is implicitly global. If a variable is assigned a new value inside a function, it is considered local. To declare it as global within the function, one must explicitly define it as such.

```
# Global variable  
x = "global value"  
  
def foo():  
    # Local variable  
    x = "local value"  
    print("Inside the function:", x)  
  
foo()  
print("Outside the function:", x)  
  
def bar():  
    global x  
    x = "modified global"  
    print("Inside the function:", x)  
  
bar()  
print("Outside the function:", x)  
  
↳ Inside the function: local value  
   Outside the function: global value  
   Inside the function: modified global  
   Outside the function: modified global
```

Q15. Do sets, dictionaries and tuples also support comprehensions?

```
# Give examples of set and dictionary comprehension
```

Q16. What is a lambda function? Using lambda function to print new list that is square of each element in first list.

```
## write your answer here  
my_list = [1,2,3,4,5]  
## write code here
```

Q17. When is not a good time to use python generators?

Ans: Use list instead of generator when:

- You need to access the data multiple times (i.e. cache the results instead of recomputing them)
- You need random access (or any access other than forward sequential order):
- You need to join strings (which requires two passes over the data)

Q18. Print the first 3 items in "fruits" list declared above? Also print the last item. (List slicing)

```
## write your code
```

Q19. Print the index of a "Banana" item in a "fruits" list?

```
# write your code here
```

Q20. Create a greet() function that accepts your name as input and then prints a welcome message.

```
## write your code
```

Q21. How can you generate random numbers in Python? Generate following numbers

- Random floating point number
- Random interger between 1 and 10
- Random floating point number between 1 and 10

```
# write your code
```

Q22. What is the purpose of is, not and in operators?

```
a = 10
b = 20
print(a is b)
print(a is not b)
```

```
list = [1, 2, 3, 4, 5]
if ( a in list ):
    print(" a is available in the given list")
else:
    print(" a is not available in the given list")
```

```
False
True
a is not available in the given list
```

Q23. What does this mean: *args, *kwargs? And why would we use it?

```
# write your answer and code
```

Q24. How are format() and f{} operations used to combine a string and a number?

```
name = "John"
age = 25
## Print name and age using both techniques
```

Q25 Explain split(), sub(), subn() methods of "re" module in Python.

Ans: To modify the strings, Python's "re" module is providing 3 methods. They are:

split() – uses a regex pattern to "split" a given string into a list.

sub() – finds all substrings where the regex pattern matches and then replace them with a different string

subn() – it is similar to sub() and also returns the new string along with the no. of replacements.

```
string = "Hi. My name is John. Nice to meet you."
print(string.split(sep='.'))
```

```
import re
string = "Hi. My name is John. Nice to meet you."
print(re.split(r'\.', string))
```

```
['Hi', ' My name is John', ' Nice to meet you', '']
['Hi', ' My name is John', ' Nice to meet you', '']
```

Q26. What advantages do NumPy arrays offer over (nested) Python lists?

write your answer

Q27 Convert the string "pYTHON pROGRAMMING" to "Python Programming" string operations?

```
str = "pYTHON pROGRAMMING"
# write your code
```

Q28. How To Assign Values For The Class Attributes At Runtime?

Ans: We need to add an init method and pass input to object constructor.

```

class Human:
    def __init__(self, profession):
        self.profession = profession
    def set_profession(self, new_profession):
        self.profession = new_profession

# write code to assign profession

```

Q29. How to map the corresponding indices of multiple containers(list/array/dict)?

```

employee = ["John", "Jane", "Jack"]
salary = [1000, 2000, 3000]
dept = ["HR", "IT", "Finance"]

# write code

```

Expected output:

John 1000 HR

Jane 2000 IT

Jack 3000 Finance

Q30 What Is the command To debug A Python Program?

Write your answer here

Q31 Given two arrays, write a python function to return the intersection of the two? For example, X = [1,2,3,4] and Y = [3,2] it should return [2,3]

```

X = [1,2,3,4]
Y = [3,2]
# write your code

```

Q32. How the concept of broadcasting is used in adding scalars to arrays in Numpy?

```

arr = np.array([1, 2, 3])
scalar = 10
# write your code

```

Q33. What is the use of the *axis* parameter in NumPy functions? Print the mean of each column in following array.

```

scores = np.array([[8, 6, 7, 9],
                   [4, 7, 6, 8],
                   [3, 5, 9, 2],
                   [4, 6, 2, 8]])
## write your code

```

Q34. How do we concatenate two arrays in Numpy using functions `concat()`, `hstack()` and `vstack()`

```

arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])

## write your code

```

Q35. Create a 3x3 identity matrix. Next create an array of 20 linearly spaced points between 0 and 1

```

## write your code

```

Q36. Sort the array below by second row. Then return an array of odd rows and even columns from below numpy array.

```

Array = np.array([[3, 6, 12, 9],
                  [34, 43, 73, 65], [82, 22, 12, 34], [53, 94, 66, 47]])

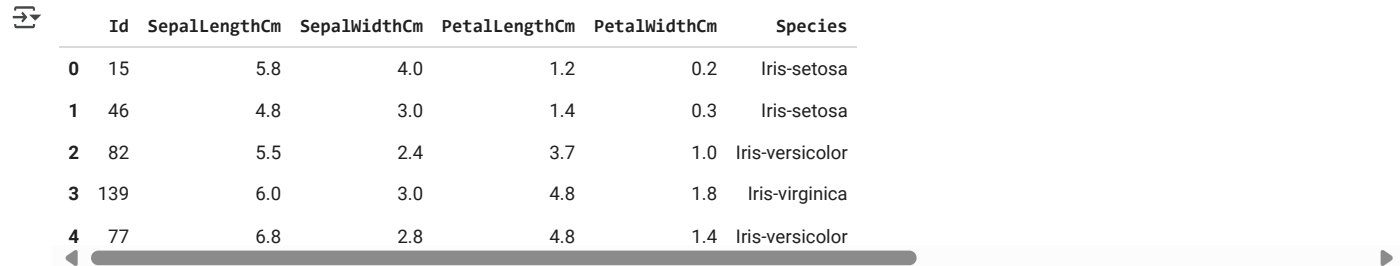
## write your code

```

✓ Pandas questions

Q37. Read the "Iris.csv" file using Pandas? Print the statistical summary of dataframe. Then replace 10 random values in column 2 to 5 with Nan values.

```
import pandas as pd
df = pd.read_csv('/content/Iris.csv')
# print the top 5 rows of dataframe
df.head()
```



	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	15	5.8	4.0	1.2	0.2	Iris-setosa
1	46	4.8	3.0	1.4	0.3	Iris-setosa
2	82	5.5	2.4	3.7	1.0	Iris-versicolor
3	139	6.0	3.0	4.8	1.8	Iris-virginica
4	77	6.8	2.8	4.8	1.4	Iris-versicolor

```
## write your code
```

Ques: How can we read directly the ".csv" data file into a numpy array.

```
# write your code
```

Q38. Check whether dataframe is empty? If not empty, iterate through each row using iterrows(), and print the rows where sepal length is less than 5.0 and species is Iris-setosa.

```
## write your code
```

Q39. Find the names of unique species. Then print number of datapoints belonging to each species. Print the average sepal length and petal width in each species

```
# write your code
```

Q40. How to check for NULL values in a dataframe? Replace the Nan values in dataframe with mean value of that column.

```
## write your code
```

Q41. Add a new column to the dataframe where value is 1 if sepal width greater than 3.5, else 0.

```
## write your code
```

Q42. Sort the dataframe on the basis of "sepal width". Print the rows with first 5 smallest petal length. (with and without sort function).

```
# write your code
```

Q43. Drop second column of dataframe and then create new dataframe with only last 10 rows of Iris data.

```
## write your code
```

Q44. Perform normalization of dataframe using max/min normalization.

```
##write your code
```

Q45. Sample the 20% dataframe with and without replacement.[link text]

```
## write your code
```

Q46. How to convert categorical data into numeric format. Convert "species" column into numerical format


```
## write your code
```

Q47. Explain the use of the loc and iloc methods.

```
## Example
```

Q48. Ice-cream flavour Counts Let's say we had everyone in class vote on their favorite ice-cream flavour by entering an ice-cream flavour into an online poll. The results were then put into a file so that each line of the file is an ice-cream flavour and the number of votes it received. Here some of the results:

Vanilla: 34 Butterscotch: 20 Chocolate: 14 Mango: 7 chocolate: 5 vanilla: 2

What we want to do is to read in the data from this file into a dictionary where the key is the ice-cream flavour (string) and the value is the number of votes this flavour got (integer).

Unfortunately, the polling mechanism didn't take into account that some people would capitalize the names of ice-cream flavour differently. Notice that above, "Chocolate" and "chocolate" each get their own score, but really we'd like to sum all the votes for each ice-cream flavour in a case-insensitive way, so that we see that 39 people have "chocolate" as their favorite ice-cream flavour.

write a helper function `get_flavour_counts(filename)` that takes in a filename and returns a dictionary with the duplicate counts (for keys with the same characters but different capitalization) combined. The new keys should be all lowercase. Calling our helper function with the file above would return:

vanilla: 36 butterscotch: 20 chocolate: 19 Mango: 7

```
def get_flavour_counts(filename):  
    # write your code  
    return counts_dict
```

Q49. Below is an example of toy data. Now, check if this data is imbalanced? What are different ways to make this toy data balanced. Implement your balancing approach. The code should be such that it can be extended to any number of classes and any number of datapoints

```
from sklearn.datasets import make_classification
```

```
X, y = make_classification(  
    n_samples=1000,  
    n_features=5,  
    n_informative=2,  
    n_redundant=0,  
    n_repeated=0,  
    n_classes=2,  
    weights=[0.9, 0.1],  
    random_state=42  
)  
X,y
```

Q50. Two lists are given below, first list contains count of fruits and second list contains fruits. Create a dictionary to map count of fruits with fruit names.

```
list1 = [12, 3, 40, 5]  
list2 = ["apples", "banana", "cherry", "grapes"]
```

Start coding or [generate](#) with AI.