Introduction to programming for data science

STAT 201

Arvind Krishna

2022-09-20

Table of contents

Preface

This book is currently being written for the course STAT201.

1 Introduction to Jupyter Notebooks and programming in python

This chapter is a very brief introduction to python and Jupyter notebooks. We only discuss the content relevant for applying python to analyze data.

1.1 Installation

Anaconda: If you are new to python, we recommend downloading the Anaconda installer and following the instructions for installation. Once installed, we'll use the Jupyter Notebook interface to write code.

Quarto: We'll use Quarto to publish the **.ipynb* file containing text, python code, and the output. Download and install Quarto from here.

1.2 Jupyter notebook

1.2.1 Introduction

Jupyter notebook is an interactive platform, where you can write code and text, and make visualizations. You can access Jupyter notebook from the Anaconda Navigator, or directly open the Jupyter Notebook application itself. It should automatically open up in your default browser. The figure below shows a Jupyter Notebook opened with Google Chrome. This page is called the *landing page* of the notebook.

```
<IPython.core.display.Image object>
```

To create a new notebook, click on the `New` button and select the `Python 3` option. You should see a blank notebook as in the figure below.

```
<IPython.core.display.Image object>
```

1.2.2 Writing and executing code

Code cell: By default, a cell is of type *Code*, i.e., for typing code, as seen as the default choice in the dropdown menu below the *Widgets* tab. Try typing a line of python code (say, `2+3`) in an empty code cell and execute it by pressing *Shift+Enter*. This should execute the code, and create an new code cell. Pressing *Ctlr+Enter for Windows (or Cmd+Enter for Mac)* will execute the code without creating a new cell.

Commenting code in a code cell: Comments should be made while writing the code to explain the purpose of the code or a brief explanation of the tasks being performed by the code. A comment can be added in a code cell by preceding it with a # sign. For example, see the comment in the code below.

Writing comments will help other users understand your code. It is also useful for the coder to keep track of the tasks being performed by their code.

```
#This code adds 3 and 5
3+5
```

```
8
```

Markdown cell: Although a comment can be written in a code cell, a code cell cannot be used for writing headings/sub-headings, and is not appropriate for writing lengthy chunks of text. In such cases, change the cell type to *Markdown* from the dropdown menu below the *Widgets* tab. Use any markdown cheat sheet found online, for example, this one to format text in the markdown cells.

Give a name to the notebook by clicking on the text, which says 'Untitled'.

1.2.3 Saving and loading notebooks

Save the notebook by clicking on `File`, and selecting `Save as`, or clicking on the `Save and Checkpoint` icon (below the `File` tab). Your notebook will be saved as a file with an exptension *ipynb*. This file will contain all the code as well as the outputs, and can be loaded and edited by a Jupyter user. To load an existing Jupyter notebook, navigate to the folder of the notebook on the *landing page*, and then click on the file to open it.

1.2.4 Rendering notebook as HTML

We'll use Quarto to print the **.ipynb* file as HTML. Check the procedure for rendering a notebook as HTML here. You have several options to format the file.

You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`.

1.3 In-class exercise

1. Create a new notebook.
2. Save the file as `In_class_exercise_1`.
3. Give a heading to the file - `First HTML file`.
4. Print `Today is day 1 of my programming course`.
5. Compute and print the number of seconds in a day.

The HTML file should look like the picture below.

```
<IPython.core.display.Image object>
```

1.4 Python libraries

There are several built-in functions in python like `print()`, `abs()`, `max()`, `sum()` etc., which do not require importing any library. However, these functions will typically be insufficient for a analyzing data. Some of the popular libraries in data science and their primary purposes are as follows:

1. NumPy: Performing numerical operations and efficiently storing numerical data.
2. Pandas: Reading, cleaning and manipulating data.
3. Matplotlib, Seaborn: Visualizing data.
4. SciPy: Performing scientific computing such as solving differential equations, optimization, statistical tests, etc.
5. Scikit-learn: Data pre-processing and machine learning, with a focus on prediction.
6. Statsmodels: Developing statistical models with a focus on inference

A library can be imported using the `import` keyword. For example, a NumPy library can be imported as:

```
import numpy as np
```

Using the `as` keyboard, the NumPy library has been given the name `np`. All the functions and attributes of the library can be called using the *'np.'* prefix. For example, let us generate a sequence of whole numbers upto `10` using the NumPy function arange():

```
np.arange(8)
```

```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

Generating random numbers is very useful in python for performing simulations (we'll see in later chapters). The library random is used to generate random numbers such as integers, real numbers based on different probability distributions, etc.

Below is an example of using the `randint()` function of the library for generating random numbers in [a, b], where `a` and `b` are integers.

```
import random as rm
rm.randint(5,10) #This will generate a random number in [5,10]
```

```
7
```

1.5 Debugging and errors

Read sections 1.3 - 1.6 from http://openbookproject.net/thinkcs/python/english3e/way_of_the_program.html

1.6 Terms used in programming

Read section 1.11 from http://openbookproject.net/thinkcs/python/english3e/way_of_the_program.html

2 Data structures

In this chapter we'll learn about the python data structures that are often used or appear while analyzing data.

2.1 Tuple

Tuple is a sequence of python objects, with two key characeterisics: (1) the number of objects are fixed, and (2) the objects are immutable, i.e., they cannot be changed.

Tuple can be defined as a sequence of python objects separated by commas, and enclosed in rounded brackets (). For example, below is a tuple containing three integers.

```
tuple_example = (2,7,4)
```

We can check the data type of a python object using the *type()* function. Let us check the data type of the object *tuple_example*.

```
type(tuple_example)
```

```
tuple
```

Elements of a tuple can be extracted using their index within square brackets. For example the second element of the tuple *tuple_example* can be extracted as follows:

```
tuple_example[1]
```

```
7
```

Note that an element of a tuple cannot be modified. For example, consider the following attempt in changing the second element of the tuple *tuple_example*.

```
tuple_example[1] = 8
```

```
TypeError: 'tuple' object does not support item assignment
```

The above code results in an error as tuple elements cannot be modified.

2.1.1 Practice exercise 1

USA's GDP per capita from 1960 to 2021 is given by the tuple T in the code cell below. The values are arranged in ascending order of the year, i.e., the first value is for 1960, the second value is for 1961, and so on. Print the years in which the GDP per capita of the US increased by more than 10%.

```
T = (3007, 3067, 3244, 3375,3574, 3828, 4146, 4336, 4696, 5032,5234,5609,6094,6726,7226,78
```

Solution:

```
#Iterating over each element of the tuple
for i in range(len(T)-1):

    #Computing percentage increase in GDP per capita in the (i+1)th year
    increase = (T[i+1]-T[i])/T[i]

    #Printing the year if the increase in GDP per capita is more than 10%
    if increase>0.1:
        print(i+1961)
```

```
1973
1976
1977
1978
1979
1981
1984
```

2.1.2 Concatenating tuples

Tuples can be concatenated using the + operator to produce a longer tuple:

```
(2,7,4) + ("another", "tuple") + ("mixed","datatypes",5)
```

```
(2, 7, 4, 'another', 'tuple', 'mixed', 'datatypes', 5)
```

Multiplying a tuple by an integer results in repetition of the tuple:

```
(2,7,"hi") * 3
```

```
(2, 7, 'hi', 2, 7, 'hi', 2, 7, 'hi')
```

2.1.3 Unpacking tuples

If tuples are assigned to an expression containing multiple variables, the tuple will be unpacked and each variable will be assigned a value as per the order in which it appears. See the example below.

```
x,y,z  = (4.5, "this is a string", (("Nested tuple",5)))
```

```
x
```

```
4.5
```

```
y
```

```
'this is a string'
```

```
z
```

```
('Nested tuple', 5)
```

If we are interested in retrieving only some values of the tuple, the expression *_ can be used to discard the other values. Let's say we are interested in retrieving only the first and the last two values of the tuple:

```
x,*_,y,z  = (4.5, "this is a string", (("Nested tuple",5)),"99",99)
```

```
x
```

```
4.5
```

```
y
```

```
'99'
```

```
z
```

```
99
```

2.1.4 Practice exercise 2

USA's GDP per capita from 1960 to 2021 is given by the tuple T in the code cell below. The values are arranged in ascending order of the year, i.e., the first value is for 1960, the second value is for 1961, and so on.

Write a function that has two parameters:

1. Year : which indicates the year from which the GDP per capita are available in the second parameter
2. Tuple of GDP per capita's: Tuple consisting of GDP per capita for consecutive years starting from the year mentioned in the first parameter.

The function should return a tuple of length two, where the first element of the tuple is the number of years when the increase in GDP per capita was more than 5%, and the second element is the most recent year in which the GDP per capita increase was more than 5%.

Call the function to find the number of years, and the most recent year in which the GDP per capita increased by more than 5%, since the year 2000. Assign the `number of years` returned by the function to a variable named `num_years`, and assign the most recent year to a variable named `recent_year`. Print the values of `num_years` and `recent_year`.

```
T = (3007, 3067, 3244, 3375,3574, 3828, 4146, 4336, 4696, 5032,5234,5609,6094,6726,7226,78
```

Solution:

```
def gdp_inc(year,gdp_tuple):
    count=0
    for i in range(len(gdp_tuple)-1):

        #Computing the increase in GDP per capita for the (i+1)th year
        increase = (gdp_tuple[i+1]-gdp_tuple[i])/gdp_tuple[i]
        if increase>0.05:
            print(year+i)
```

```
            #Over-writing the value of recent_year if the increase in GDP per capita for a
            recent_year = year+i+1

            #Counting the number of years for which the increase in GDP per capita is more
            count = count+1
    return((count,recent_year))

num_years, recent_year = gdp_inc(2000,T[40:])
print("Number of years when increase in GDP per capita was more than 5% = ", num_years)
print("The most recent year in which the increase in GDP per capita was more than 5% = ",
```

```
2003
2004
2020
Number of years when increase in GDP per capita was more than 5% =  3
The most recent year in which the increase in GDP per capita was more than 5% =  2021
```

```
T = (3007, 3067, 3244, 3375,3574, 3828, 4146, 4336, 4696, 5032,5234,5609,6094,6726,7226,78
```

2.1.5 Tuple methods

A couple of useful tuple methods are `count`, which counts the occurences of an element in the tuple and `index`, which returns the position of the first occurance of an element in the tuple:

```
tuple_example = (2,5,64,7,2,2)
```

```
tuple_example.count(2)
```

```
3
```

```
tuple_example.index(2)
```

```
0
```

Now that we have an idea about tuple, let us try to think where it can be used.

```
<IPython.core.display.HTML object>
```

2.2 List

List is a sequence of python objects, with two key characeterisics that differentiates it from tuple: (1) the number of objects are variable, i.e., objects can be added or removed from a list, and (2) the objects are mutable, i.e., they can be changed.

List can be defined as a sequence of python objects separated by commas, and enclosed in square brackets []. For example, below is a list consisting of three integers.

```
list_example = [2,7,4]
```

2.2.1 Adding and removing elements in a list

We can add elements at the end of the list using the *append* method. For example, we append the string 'red' to the list *list_example* below.

```
list_example.append('red')
```

```
list_example
```

```
[2, 7, 4, 'red']
```

Note that the objects of a list or a tuple can be of different datatypes.

An element can be added at a specific location of the list using the *insert* method. For example, if we wish to insert the number 2.32 as the second element of the list *list_example*, we can do it as follows:

```
list_example.insert(1,2.32)
```

```
list_example
```

```
[2, 2.32, 7, 4, 'red']
```

For removing an element from the list, the *pop* and *remove* methods may be used. The *pop* method removes an element at a particular index, while the *remove* method removes the element's first occurence in the list by its value. See the examples below.

Let us say, we need to remove the third element of the list.

```
list_example.pop(2)
```

```
7
```

```
list_example
```

```
[2, 2.32, 4, 'red']
```

Let us say, we need to remove the element 'red'.

```
list_example.remove('red')
```

```
list_example
```

```
[2, 2.32, 4]
```

```
#If there are multiple occurences of an element in the list, the first occurence will be r
list_example2 = [2,3,2,4,4]
list_example2.remove(2)
list_example2
```

```
[3, 2, 4, 4]
```

For removing multiple elements in a list, either `pop` or `remove` can be used in a `for` loop, or a `for` loop can be used with a condition. See the examples below.

Let's say we need to remove integers less than 100 from the following list.

```
list_example3 = list(range(95,106))
list_example3
```

```
[95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105]
```

```
#Method 1: For loop with remove
list_example3_filtered = list(list_example3) #
```

```
for element in list_example3:
    if element<100:
        list_example3_filtered.remove(element)
print(list_example3_filtered)
```

```
[100, 101, 102, 103, 104, 105]
```

Q1: What's the need to define a new variable `list\_example3\_filtered` in the above code?

A1: Replace list_example3_filtered with list_example3 and identify the issue.

```
#Method 2: For loop with condition
[element for element in list_example3 if element>100]
```

```
[101, 102, 103, 104, 105]
```

2.2.2 List comprehensions

List comprehension is a compact way to create new lists based on elements of an existing list or other objects.

Example: Create a list that has squares of natural numbers from 5 to 15.

```
sqrt_natural_no_5_15 = [(x**2) for x in range(5,16)]
print(sqrt_natural_no_5_15)
```

```
[25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225]
```

Example: Create a list of tuples, where each tuple consists of a natural number and its square, for natural numbers ranging from 5 to 15.

```
sqrt_natural_no_5_15 = [(x,x**2) for x in range(5,16)]
print(sqrt_natural_no_5_15)
```

```
[(5, 25), (6, 36), (7, 49), (8, 64), (9, 81), (10, 100), (11, 121), (12, 144), (13, 169), (14
```

2.2.3 Practice exercise 3

Below is a list consisting of responses to the question: "At what age do you think you will marry?" from students of the STAT303-1 Fall 2022 class.

```
exp_marriage_age=['24','30','28','29','30','27','26','28','30+','26','28','30','30','30','
```

Use list comprehension to:

2.2.3.1

Remove the elements that are not integers - such as *'probably never', '30+'*, etc. What is the length of the new list?

Hint: The built-in python function of the `str` class - isdigit() may be useful to check if the string contains only digits.

```
exp_marriage_age_num = [x for x in exp_marriage_age if x.isdigit()==True]
print("Length of the new list = ",len(exp_marriage_age_num))
```

```
Length of the new list =  181
```

2.2.3.2

Cap the values greater than 80 to 80, in the clean list obtained in (1). What is the mean age when people expect to marry in the new list?

```
exp_marriage_age_capped = [min(int(x),80) for x in exp_marriage_age_num]
print("Mean age when people expect to marry = ", sum(exp_marriage_age_capped)/len(exp_marr
```

```
Mean age when people expect to marry =  28.955801104972377
```

2.2.3.3

Determine the percentage of people who expect to marry at an age of 30 or more.

```
print("Percentage of people who expect to marry at an age of 30 or more =", str(100*sum([1
```

```
Percentage of people who expect to marry at an age of 30 or more = 37.01657458563536 %
```

2.2.4 Concatenating lists

As in tuples, lists can be concatenated using the + operator:

```
import time as tm
```

```
list_example4 = [5,'hi',4]
list_example4 = list_example4 + [None,'7',9]
list_example4
```

```
[5, 'hi', 4, None, '7', 9]
```

For adding elements to a list, the `extend` method is preferred over the `+` operator. This is because the `+` operator creates a new list, while the `extend` method adds elements to an existing list. Thus, the `extend` operator is more memory efficient.

```
list_example4 = [5,'hi',4]
list_example4.extend([None, '7', 9])
list_example4
```

```
[5, 'hi', 4, None, '7', 9]
```

2.2.5 Sorting a list

A list can be sorted using the `sort` method:

```
list_example5 = [6,78,9]
list_example5.sort(reverse=True) #the reverse argument is used to specify if the sorting i
list_example5
```

```
[78, 9, 6]
```

2.2.6 Slicing a list

We may extract or update a section of the list by passing the starting index (say `start`) and the stopping index (say `stop`) as `start:stop` to the index operator []. This is called *slicing* a list. For example, see the following example.

```
list_example6 = [4,7,3,5,7,1,5,87,5]
```

Let us extract a slice containing all the elements from the the 3rd position to the 7th position.

```
list_example6[2:7]
```

```
[3, 5, 7, 1, 5]
```

Note that while the element at the `start` index is included, the element with the `stop` index is excluded in the above slice.

If either the `start` or `stop` index is not mentioned, the slicing will be done from the beginning or until the end of the list, respectively.

```
list_example6[:7]
```

```
[4, 7, 3, 5, 7, 1, 5]
```

```
list_example6[2:]
```

```
[3, 5, 7, 1, 5, 87, 5]
```

To slice the list relative to the end, we can use negative indices:

```
list_example6[-4:]
```

```
[1, 5, 87, 5]
```

```
list_example6[-4:-2:]
```

```
[1, 5]
```

An extra colon (':') can be used to slice every ith element of a list.

```
#Selecting every 3rd element of a list
list_example6[::3]
```

```
[4, 5, 5]
```

```
#Selecting every 3rd element of a list from the end
list_example6[::-3]
```

```
[5, 1, 3]
```

```
#Selecting every element of a list from the end or reversing a list
list_example6[::-1]
```

```
[5, 87, 5, 1, 7, 5, 3, 7, 4]
```

2.2.7 Practice exercise 4

Start with the list [8,9,10]. Do the following:

2.2.7.1

Set the second entry (index 1) to 17

```
L = [8,9,10]
L[1]=17
```

2.2.7.2

Add 4, 5, and 6 to the end of the list

```
L.extend([4, 5, 6])
```

2.2.7.3

Remove the first entry from the list

```
L.pop(0)
```

```
8
```

2.2.7.4

Sort the list

```
L.sort()
```

2.2.7.5

Double the list (concatenate the list to itself)

```
L.extend(L)
```

2.2.7.6

Insert 25 at index 3

The final list should equal [4,5,6,25,10,17,4,5,6,10,17]

```
L.insert(3, 25)
```

```
L
```

```
[4, 5, 6, 25, 10, 17, 4, 5, 6, 10, 17]
```

Now that we have an idea about lists, let us try to think where it can be used.

```
<IPython.core.display.HTML object>
```

Now that we have learned about lists and tuples, let us compare them.

Q2: A list seems to be much more flexible than tuple, and can replace a tuple almost everywhere. Then why use tuple at all?

A2: The additional flexibility of a list comes at the cost of efficiency. Some of the advatages of a tuple over a list are as follows:

1. Since a list can be extended, space is over-allocated when creating a list. A tuple takes less storage space as compared to a list of the same length.

2. Tuples are not copied. If a tuple is assigned to another tuple, both tuples point to the same memory location. However, if a list is assigned to another list, a new list is created consuming the same memory space as the orignial list.
3. Tuples refer to their element directly, while in a list, there is an extra layer of pointers that refers to their elements. Thus it is faster to retrieve elements from a tuple.

The examples below illustrate the above advantages of a tuple.

```
#Example showing tuples take less storage space than lists for the same elements
tuple_ex = (1, 2, 'Obama')
list_ex = [1, 2, 'Obama']
print("Space taken by tuple =",tuple_ex.__sizeof__()," bytes")
print("Space taken by list =",list_ex.__sizeof__()," bytes")
```

```
Space taken by tuple = 48  bytes
Space taken by list = 64  bytes
```

```
#Examples showing that a tuples are not copied, while lists can be copied
tuple_copy = tuple(tuple_ex)
print("Is tuple_copy same as tuple_ex?", tuple_ex is tuple_copy)
list_copy = list(list_ex)
print("Is list_copy same as list_ex?",list_ex is list_copy)
```

```
Is tuple_copy same as tuple_ex? True
Is list_copy same as list_ex? False
```

```
#Examples showing tuples takes lesser time to retrieve elements
import time as tm
tt = tm.time()
list_ex = list(range(1000000)) #List containinig whole numbers upto 1 million
a=(list_ex[::-2])
print("Time take to retrieve every 2nd element from a list = ", tm.time()-tt)

tt = tm.time()
tuple_ex = tuple(range(1000000)) #tuple containinig whole numbers upto 1 million
a=(tuple_ex[::-2])
print("Time take to retrieve every 2nd element from a tuple = ", tm.time()-tt)
```

```
Time take to retrieve every 2nd element from a list =  0.03579902648925781
Time take to retrieve every 2nd element from a tuple =  0.02684164047241211
```

2.3 Dictionary

A dictionary consists of key-value pairs, where the keys and values are python objects. While values can be any python object, keys need to be immutable python objects, like strings, integers, tuples, etc. Thus, a list can be a value, but not a key, as elements of a list can be changed. A dictionary is defined using the keyword `dict` along with curly braces, colons to separate keys and values, and commas to separate elements of a dictionary:

```
dict_example = {'USA':'Joe Biden', 'India':'Narendra Modi', 'China':'Xi Jinping'}
```

Elements of a dictionary can be retrieved by using the corresponding key.

```
dict_example['India']
```

```
'Narendra Modi'
```

2.3.1 Adding and removing elements in a dictionary

New elements can be added to a dictionary by defining a key in square brackets and assiging it to a value:

```
dict_example['Japan'] = 'Fumio Kishida'
dict_example['Countries'] = 4
dict_example
```

```
{'USA': 'Joe Biden',
 'India': 'Narendra Modi',
 'China': 'Xi Jinping',
 'Japan': 'Fumio Kishida',
 'Countries': 4}
```

Elements can be removed from the dictionary using the `del` method or the `pop` method:

```
#Removing the element having key as 'Countries'
del dict_example['Countries']
```

```
dict_example
```

```
{'USA': 'Joe Biden',
 'India': 'Narendra Modi',
 'China': 'Xi Jinping',
 'Japan': 'Fumio Kishida'}
```

```
#Removing the element having key as 'USA'
dict_example.pop('USA')
```

```
'Joe Biden'
```

```
dict_example
```

```
{'India': 'Narendra Modi', 'China': 'Xi Jinping', 'Japan': 'Fumio Kishida'}
```

New elements can be added, and values of exisiting keys can be changed using the `update` method:

```
dict_example = {'USA':'Joe Biden', 'India':'Narendra Modi', 'China':'Xi Jinping','Countrie
dict_example
```

```
{'USA': 'Joe Biden',
 'India': 'Narendra Modi',
 'China': 'Xi Jinping',
 'Countries': 3}
```

```
dict_example.update({'Countries':4, 'Japan':'Fumio Kishida'})
```

```
dict_example
```

```
{'USA': 'Joe Biden',
 'India': 'Narendra Modi',
 'China': 'Xi Jinping',
 'Countries': 4,
 'Japan': 'Fumio Kishida'}
```

2.3.2 Iterating over elements of a dictionary

The items() attribute of a dictionary can be used to iterate over elements of a dictionary.

```
for key,value in dict_example.items():
    print("The Head of State of",key,"is",value)
```

```
The Head of State of USA is Joe Biden
The Head of State of India is Narendra Modi
The Head of State of China is Xi Jinping
The Head of State of Countries is 4
The Head of State of Japan is Fumio Kishida
```

2.3.3 Practice exercise 5

The GDP per capita of USA for most years from 1960 to 2021 is given by the dictionary `D` given in the code cell below.

Find:

1. The GDP per capita in 2015
2. The GDP per capita of 2014 is missing. Update the dictionary to include the GDP per capita of 2014 as the average of the GDP per capita of 2013 and 2015.
3. Impute the GDP per capita of other missing years in the same manner as in (2), i.e., as the average GDP per capita of the previous year and the next year. Note that the GDP per capita is not missing for any two consecutive years.
4. Print the years and the imputed GDP per capita for the years having a missing value of GDP per capita in (3).

```
D = {'1960':3007,'1961':3067,'1962':3244,'1963':3375,'1964':3574,'1965':3828,'1966':4146,'
```

Solution:

```
print("GDP per capita in 2015 =", D['2015'])
D['2014'] = (D['2013']+D['2015'])/2

#Iterating over all years from 1960 to 2021
for i in range(1960,2021):

    #Imputing the GDP of the year if it is missing
    if str(i) not in D.keys():
        D[str(i)] = (D[str(i-1)]+D[str(i+1)])/2
        print("Imputed GDP per capita for the year",i,"is $",D[str(i)])
```

```
GDP per capita in 2015 = 56763
Imputed GDP per capita for the year 1969 is $ 4965.0
Imputed GDP per capita for the year 1977 is $ 9578.5
Imputed GDP per capita for the year 1999 is $ 34592.0
```

2.3.4 Practice exercise 6

The object `deck` defined below corresponds to a deck of cards. Estimate the probablity that a five card hand will be a flush, as follows:

1. Write a function that accepts a hand of 5 cards as argument, and returns whether the hand is a flush or not.
2. Randomly pull a hand of 5 cards from the deck. Call the function developed in (1) to determine if the hand is a flush.
3. Repeat (2) 10,000 times.
4. Estimate the probability of the hand being a flush from the results of the 10,000 simulations.

You may use the function shuffle() from the `random` library to shuffle the deck everytime before pulling a hand of 5 cards.

```
deck = [{'value':i, 'suit':c}
for c in ['spades', 'clubs', 'hearts', 'diamonds']
for i in range(2,15)]
```

Solution:

```
import random as rm

#Function to check if a 5-card hand is a flush
def chck_flush(hands):

    #Assuming that the hand is a flush, before checking the cards
    yes_flush =1

    #Storing the suit of the first card in 'first_suit'
    first_suit = hands[0]['suit']

    #Iterating over the remaining 4 cards of the hand
    for j in range(1,len(hands)):

        #If the suit of any of the cards does not match the suit of the first card, the ha
```

```
            if first_suit!=hands[j]['suit']:
                yes_flush = 0;

                #As soon as a card with a different suit is found, the hand is not a flush and
                break;
        return yes_flush

    flush=0
    for i in range(10000):

        #Shuffling the deck
        rm.shuffle(deck)

        #Picking out the first 5 cards of the deck as a hand and checking if they are a flush
        #If the hand is a flush it is counted
        flush=flush+chck_flush(deck[0:5])

    print("Probability of obtaining a flush=", 100*(flush/10000),"%")
```

```
Probability of obtaining a flush= 0.26 %
```

2.4 Practice exercise 7

The code cell below defines an object having the nutrition information of drinks in starbucks. Assume that the manner in which the information is structured is consistent throughout the object.

```
, 'value': 1}, {'Nutrition_type': 'Sodium', 'value': 10}], 'Featured Dark Roast'
starbucks_drinks_nutrition={'Cool Lime Starbucks Refreshers Beverage': [{'Nutrition_type': 'Cal
```

Use the object above to answer the following questions:

2.4.1

What is the datatype of the object?

2.4.1.1

If the object in (1) is a dictionary, what is the datatype of the values of the dictionary?

2.4.1.2

If the object in (1) is a dictionary, what is the datatype of the elements within the values of the dictionary?

2.4.1.3

How many calories are there in `Iced Coffee`?

2.4.1.4

Which drink(s) have the highest amount of protein in them, and what is that protein amount?

2.4.1.5

Which drink(s) have a fat content of more than 10g, and what is their fat content?

3 R: Recursion

3.1 Recursion

Recursion is a method of solving a problem by dividing it into smaller instances of the same problem. Recursion solves such problems by using functions that call themselves from within their own code. This forms a loop, where every time the function is called, it calls itself again and again. However, every time the function calls itself, it checks certain condition(s) which are the stopping condition(s). When such condition(s) are true the function will stop calling itself. These conditions are called the base case of the recursive function.

Every recursive function must have at least two cases:

1. Base case: This is the simplest case that can be answered directly, and the function does not call itself.

2. Recursive case: This is a relatively more complex case that cannot be answered directly, but can be described as a smaller instance of the same problem. In this case, the function calls itself to answer the smaller problem.

Below is an example, where we defined a function that computes the factorial of an integer by recursion.

```
factorial<-function(n)
{
  if(n==1)      #Base case
  {
      return(1)
  }
  return(n*factorial(n-1)) #Recursive case
}
factorial(5)
```

```
[1] 120
```

In the above example, the case $n = 1$ is the base case, where the function does not need to call itself, and returns 1. All other cases, where $n > 1$, and $n \in \mathbb{Z}$ are recursive cases, where the function calls itself with a smaller instance of the same problem.

A recursive function must satisfy the following conditions:

1. There must be a case for all valid inputs.
2. There must be a base case that makes no recursive calls.
3. When the function makes a recursive call, it should be to a simpler instance and make forward progress towards the base case.

Example: Write a recursive function that returns the n^{th} term of the Fibonacci sequence, where n is an integer, and $n > 0$. In a Fibonacci sequence, each number is the sum of the preceding two numbers, and the sequence starts from $0, 1$. The sequence is as follows:

$0, 1, 1, 2, 3, 5, 8, 13, 21, ...$

```
fibonacci<-function(n)
{
  if(n==0 | n==1){  #Base case
    return(n)
  }
  return(fibonacci(n-1)+fibonacci(n-2))  #Recursive case
}
#The function `fibonacci` prints the n+1th term of the fibonacci sequence when `n` is pass
nth_term<-function(N)
{
  fibonacci(N-1)
}
nth_term(7)
```

```
[1] 8
```

3.1.1 Practice exercise 1

Write a recursive function that computes the sum of squares of the first N natural numbers, where N is a parameter to the function.

```
squares<-function(N)
{
  if(N==1)  #Base case
  {
```

```
    return(1)
  }else{    #Recursive case
    return(N**2+squares(N-1))
  }
}
squares(10)
```

3.1.2 Practice exercise 2

Write a function that counts the occurrence of digit k in a given integer n using recursion. The function has n and k as parameters.

```
freq_digits<-function(n,d)
{
  if(n==0)
  {
    return(0)
  }
  digit = n%%10
  n_int = as.integer(n/10)
  if(digit==d)
  {
    return(1+freq_digits(n_int,d))
  }
  return(freq_digits(n_int,d))
}
freq_digits(8670800,0)
```

3.1.3 Practice exercise 3

Use recursion to write a function that accepts a word as an argument, and returns `TRUE` if the word is a palindrome, otherwise returns `FALSE`.

```
word<-'racecar'
palindrome<-function(word)
{
  if(nchar(word)<=1)
  {
    return(TRUE)
```

```
  }else if(substr(word,1,1)==substr(word,nchar(word),nchar(word)))
  {
    palindrome(substr(word,2,nchar(word)-1))
  }else{
    return(FALSE)
  }
}
palindrome(word)
```

3.2 Recursion vs iteration

Recursion is typically used when the problem is naturally recursive (for e.g., generating a Fibonacci sequence), or the data is naturally recursive (for e.g., filesystem). Recursive solutions can be easy to read and understand as compared to the corresponding iterative solution.

One downside of recursion is that it may take more space than an iterative solution. Building up a stack of recursive calls consumes memory temporarily, and the stack is limited in size, which may become a limit on the size of the problem that the recursive implementation can solve.

A Assignment A

Instructions

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. There are 5 points for clealiness and organization. The breakdow is as follows:

- Must be an HTML file rendered using Quarto (1.5 pts).
- There aren't excessively long outputs of extraneous information (e.g. no printouts of unnecessary results without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)
- The code should be commented and clearly written with intuitive variable names. For example, use variable names such as number_input, factor, hours, instead of a,b,xyz, etc. (1.5 pts)

6. The assignment is worth 100 points, and is due on **13th April 2023 at 11:59 pm**.

A.1 Alarm clock

A.1.1 When does the alarm go off?

You look at the clock and it is exactly 2pm. You set an alarm to go off in 510 hours. At what time does the alarm go off? If the answer is say, 4 pm, then your code should print - `"The alarm goes off at 4 pm"`.

(2 points)

A.1.2 User-friendly alarm clock

Write a program to solve the general version of the above problem. Ask the user for - (1) the time now (in hours), and (2) the number of hours for the alarm to go off. Your program should output the time at which the alarm goes off. Both the user inputs must be in {0, 1, 2..., 22, 23}. If the answer is, say 14:00 hours, then your program should print - "The alarm goes off at 14:00 hours.

Show the output of your program when the user inputs `7` as the current time, and `95` as the number of hours for the alarm to go off.

(4 points)

A.2 Finding prime factors

A.2.1 Prime or not

Write a program that checks if a positive integer is prime or not. Show the output when the program is used to check if `89` is prime or not.

(2 points)

A.2.2 Factors

Prompt the user to input a positive integer. Write a program that prints the **factors** of the positive integer input by the user. Show the output of the program if the user inputs `190`.

(2 points)

A.2.3 Prime factors

Prompt the user to input a positive integer. Update the program in 2(b) to print the **prime factors** of the positive integer input by the user. Show the output of the program if the user inputs `190`.

(8 points)

A.2.4 User-friendly prime factor calculator

Update the program in 2(c), so that it prints "Incorrect input, please enter positive integer" if the user does not enter a positive integer, and then prompts the user to input a positive integer. The program should continue to prompt the user to enter a positive integer until the user successfully enters a positive integer. Show the output of the program if the user enters `"seventy"` in the first attempt, `"#70"` in the second attempt, and `70` in the third attempt.

(12 points)

A.3 Number of words in a sentence

Prompt the user to input an english sentence. Write a program that counts and prints the number of words in the sentence input by the user. The program should continue to run until the user inputs the sentence - "end program". Show the output of the program if the user enters `"this is the time to sleep"` in the first attempt, `"this is too much work for a day"` in the second attempt, and `"end program"` in the third attempt.

Hint: Count the number of spaces

(10 points)

A.4 Survival of rabbits

In many environments, two or more species compete for the available resources. Classic predator–prey equations have been used to simulate or predict the dynamics of biological systems in which two species interact, one as a predator and the other as prey. You will use a simplified version of the Lotka-Volterra equations for modeling fox/rabbit populations, described below.

Let the following variables be defined as:

r_t: The number of prey (rabbits) at time t, where t corresponds to a certain year.

f_t: The number of predators (foxes) at time t, where t corresponds to a certain year.

α: The birth rate of prey.

β: The death rate of prey (depends on predator population).

γ: The birth rate of predators (depends on prey population).

δ: The death rate of predators.

Then, we can define the populations of the next time period or the next year $(t+1)$ using the following system of equations:

$$r_{t+1} = r_t + \alpha r_t - \beta r_t f_t,$$

$$f_{t+1} = f_t + \gamma f_t r_t - \delta f_t$$

A.4.1 Number of rabbits and foxes

Write a program that uses the following parameter values, and calculates and prints the populations of the rabbits and foxes for each year upto the next 14 years. Since the number of rabbits and foxes cannot be floating-point numbers, use the in-built python function `round()` to round-off the calculated values to integers. Also, we cannot have negative rabbits or negative foxes, so if the population values are ever negative, consider the population to be zero instead.

$r_0 = 500$

$f_0 = 1$

$\alpha = 0.2$

$\beta = 0.005$

$\gamma = 0.001$

$\delta = 0.2$

The output of the program should be as follows:

```
At time t = 0, there are 500 rabbits, and 1 foxes

At time t = 1, there are 598 rabbits, and 1 foxes

At time t = 2, there are 713 rabbits, and 2 foxes

At time t = 3, there are 849 rabbits, and 3 foxes

At time t = 4, there are 1007 rabbits, and 5 foxes

At time t = 5, there are 1186 rabbits, and 8 foxes
```

```
At time t = 6, there are 1375 rabbits, and 16 foxes
At time t = 7, there are 1538 rabbits, and 35 foxes
At time t = 8, there are 1573 rabbits, and 83 foxes
At time t = 9, there are 1237 rabbits, and 196 foxes
At time t = 10, there are 270 rabbits, and 400 foxes
At time t = 11, there are 0 rabbits, and 428 foxes
At time t = 12, there are 0 rabbits, and 342 foxes
At time t = 13, there are 0 rabbits, and 274 foxes
At time t = 14, there are 0 rabbits, and 219 foxes
```

(10 points)

A.4.2 How long can 100 rabbits survive?

Suppose at $t = 0$, there are 100 rabbits, i.e., $r_0 = 100$. How many foxes should be there at $t = 0$ (i.e., what should be f_0), such that the rabbit species survives (i.e., $r_{t\_max} > 0$) for the maximum possible number of years (t_{max}) before becoming extinct (i.e., $r_{t\_max+1} = 0$). Also, find the maximum possible number of years (i.e., t_{max}) the rabbit species will survive.

Modify the program in the previous question to compute the answers to the above questions, and print the following statement, with the blanks filled:

```
If there are ___ foxes at t = 0, the rabbit species will survive for ___
years, which is the maximum possible number of years they can survive.
```

Note: Use the same values of α, β, γ, and δ as in the previous question.

Hint:

1. Consider values of f_0 starting from 1, and upto a large number, say 1000.
2. For each value of f_0, find the number of years for which the rabbit species survives.
3. Find the value of f_0 and t for which the rabbit species surives the maximum number of years, i.e., $t = t_{max}$.

(20 points)

A.4.3 Saving rabbits from extinction

What must be the minimum number of rabbits, and the corresponding number of foxes at $t = 0$, such that the rabbit and fox species never become extinct.

Note: Use the same values of α, β, γ, and δ as in the previous question.

Hint:

1. Consider $r_0 = 1$, and then keep increasing r_0 by 1 if it's not possible for the rabbit species to survive with the value of r_0 under consideration.
2. For each r_0, consider number of foxes starting from $f_0 = 1$, and upto a large number, say $f_0 = 200$.
3. As soon as you find a combination of r_0 and f_0, such that there is no change in r_t and f_t for 2 consecutive years, you have found the values of r_0 and f_0, such that both the species maintain their numbers and never become extinct. At this point, print the result, and stop the program *(break out of all loops).*

Modify the program in the previous question to answer the above question, and print the following statement with the blanks filled:

```
For ___ foxes, and ___ rabbits at t = 0, the fox and rabbit species will
never be extinct.
```

(25 points)

B Assignment A

Instructions

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. There are 5 points for clealiness and organization. The breakdow is as follows:

- Must be an HTML file rendered using Quarto (1.5 pts).
- There aren't excessively long outputs of extraneous information (e.g. no printouts of unnecessary results without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)
- The code should be commented and clearly written with intuitive variable names. For example, use variable names such as number_input, factor, hours, instead of a,b,xyz, etc. (1.5 pts)

6. The assignment is worth 100 points, and is due on **13th April 2023 at 11:59 pm**.

B.1 Alarm clock

B.1.1 When does the alarm go off?

You look at the clock and it is exactly 2pm. You set an alarm to go off in 510 hours. At what time does the alarm go off? If the answer is say, 4 pm, then your code should print - `"The alarm goes off at 4 pm"`.

(2 points)

B.1.2 User-friendly alarm clock

Write a program to solve the general version of the above problem. Ask the user for - (1) the time now (in hours), and (2) the number of hours for the alarm to go off. Your program should output the time at which the alarm goes off. Both the user inputs must be in {0, 1, 2..., 22, 23}. If the answer is, say 14:00 hours, then your program should print - "The alarm goes off at 14:00 hours.

Show the output of your program when the user inputs `7` as the current time, and `95` as the number of hours for the alarm to go off.

(4 points)

B.2 Finding prime factors

B.2.1 Prime or not

Write a program that checks if a positive integer is prime or not. Show the output when the program is used to check if `89` is prime or not.

(2 points)

B.2.2 Factors

Prompt the user to input a positive integer. Write a program that prints the **factors** of the positive integer input by the user. Show the output of the program if the user inputs `190`.

(2 points)

B.2.3 Prime factors

Prompt the user to input a positive integer. Update the program in 2(b) to print the **prime factors** of the positive integer input by the user. Show the output of the program if the user inputs `190`.

(8 points)

B.2.4 User-friendly prime factor calculator

Update the program in 2(c), so that it prints "Incorrect input, please enter positive integer" if the user does not enter a positive integer, and then prompts the user to input a positive integer. The program should continue to prompt the user to enter a positive integer until the user successfully enters a positive integer. Show the output of the program if the user enters `"seventy"` in the first attempt, `"#70"` in the second attempt, and `70` in the third attempt.

(12 points)

B.3 Number of words in a sentence

Prompt the user to input an english sentence. Write a program that counts and prints the number of words in the sentence input by the user. The program should continue to run until the user inputs the sentence - "end program". Show the output of the program if the user enters `"this is the time to sleep"` in the first attempt, `"this is too much work for a day"` in the second attempt, and `"end program"` in the third attempt.

Hint: Count the number of spaces

(10 points)

B.4 Survival of rabbits

In many environments, two or more species compete for the available resources. Classic predator–prey equations have been used to simulate or predict the dynamics of biological systems in which two species interact, one as a predator and the other as prey. You will use a simplified version of the Lotka-Volterra equations for modeling fox/rabbit populations, described below.

Let the following variables be defined as:

r_t: The number of prey (rabbits) at time t, where t corresponds to a certain year.

f_t: The number of predators (foxes) at time t, where t corresponds to a certain year.

α: The birth rate of prey.

β: The death rate of prey (depends on predator population).

γ: The birth rate of predators (depends on prey population).

δ: The death rate of predators.

Then, we can define the populations of the next time period or the next year $(t+1)$ using the following system of equations:

$$r_{t+1} = r_t + \alpha r_t - \beta r_t f_t,$$

$$f_{t+1} = f_t + \gamma f_t r_t - \delta f_t$$

B.4.1 Number of rabbits and foxes

Write a program that uses the following parameter values, and calculates and prints the populations of the rabbits and foxes for each year upto the next 14 years. Since the number of rabbits and foxes cannot be floating-point numbers, use the in-built python function `round()` to round-off the calculated values to integers. Also, we cannot have negative rabbits or negative foxes, so if the population values are ever negative, consider the population to be zero instead.

$r_0 = 500$

$f_0 = 1$

$\alpha = 0.2$

$\beta = 0.005$

$\gamma = 0.001$

$\delta = 0.2$

The output of the program should be as follows:

```
At time t = 0, there are 500 rabbits, and 1 foxes

At time t = 1, there are 598 rabbits, and 1 foxes

At time t = 2, there are 713 rabbits, and 2 foxes

At time t = 3, there are 849 rabbits, and 3 foxes

At time t = 4, there are 1007 rabbits, and 5 foxes

At time t = 5, there are 1186 rabbits, and 8 foxes
```

```
At time t = 6, there are 1375 rabbits, and 16 foxes
At time t = 7, there are 1538 rabbits, and 35 foxes
At time t = 8, there are 1573 rabbits, and 83 foxes
At time t = 9, there are 1237 rabbits, and 196 foxes
At time t = 10, there are 270 rabbits, and 400 foxes
At time t = 11, there are 0 rabbits, and 428 foxes
At time t = 12, there are 0 rabbits, and 342 foxes
At time t = 13, there are 0 rabbits, and 274 foxes
At time t = 14, there are 0 rabbits, and 219 foxes
```

(10 points)

B.4.2 How long can 100 rabbits survive?

Suppose at $t = 0$, there are 100 rabbits, i.e., $r_0 = 100$. How many foxes should be there at $t = 0$ (i.e., what should be f_0), such that the rabbit species survives (i.e., $r_{t\_max} > 0$) for the maximum possible number of years (t_{max}) before becoming extinct (i.e., $r_{t\_max+1} = 0$). Also, find the maximum possible number of years (i.e., t_{max}) the rabbit species will survive.

Modify the program in the previous question to compute the answers to the above questions, and print the following statement, with the blanks filled:

```
If there are ___ foxes at t = 0, the rabbit species will survive for ___
years, which is the maximum possible number of years they can survive.
```

Note: Use the same values of α, β, γ, and δ as in the previous question.

Hint:

1. Consider values of f_0 starting from 1, and upto a large number, say 1000.
2. For each value of f_0, find the number of years for which the rabbit species survives.
3. Find the value of f_0 and t for which the rabbit species surives the maximum number of years, i.e., $t = t_{max}$.

(20 points)

B.4.3 Saving rabbits from extinction

What must be the minimum number of rabbits, and the corresponding number of foxes at $t = 0$, such that the rabbit and fox species never become extinct.

Note: Use the same values of α, β, γ, and δ as in the previous question.

Hint:

1. Consider $r_0 = 1$, and then keep increasing r_0 by 1 if it's not possible for the rabbit species to survive with the value of r_0 under consideration.
2. For each r_0, consider number of foxes starting from $f_0 = 1$, and upto a large number, say $f_0 = 200$.
3. As soon as you find a combination of r_0 and f_0, such that there is no change in r_t and f_t for 2 consecutive years, you have found the values of r_0 and f_0, such that both the species maintain their numbers and never become extinct. At this point, print the result, and stop the program *(break out of all loops).*

Modify the program in the previous question to answer the above question, and print the following statement with the blanks filled:

```
For ___ foxes, and ___ rabbits at t = 0, the fox and rabbit species will
never be extinct.
```

(25 points)

C Assignment B

Instructions

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. There are 5 points for clealiness and organization. The breakdow is as follows:

- Must be an HTML file rendered using Quarto (1.5 pts).
- There aren't excessively long outputs of extraneous information (e.g. no printouts of unnecessary results without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)
- The code should be commented and clearly written with intuitive variable names. For example, use variable names such as number_input, factor, hours, instead of a,b,xyz, etc. (1.5 pts)

6. The assignment is worth 100 points, and is due on **Friday, 21st April 2023 at 11:59 pm**.

C.1 Sentence analysis

C.1.1 Word count

Write a function that accepts a word, and a sentence as arguments, and returns the number of times the word occurs in the sentence.

Call the function, and print the returned value if the word is "*sea*", and the sentence is "*She sells sea shells on the sea shore when the sea is calm.*" Note that this is just an example to check your function. Your function should work for any word and sentence.

(10 points)

C.1.2 Max word count

Ask the user to input a sentence. Use the function in B.1.1 to find the word that occurs the maximum number of times in the sentence. Print the word and its number of occurences. If multiple words occur the maximum number of times, then you can print any one of them.

Check your program when the user inputs the sentence, "*She sells sea shells on the sea shore when the sea is calm.*". Your program must print, "*The word with the maximum number of occurences is 'sea' and it occurs 3 times.*" Note that this is just an example to check your program. Your program must work for any sentence.

(20 points)

C.2 Prime factors

C.2.1 Prime

Write a function that checks if an integer is prime. The function must accept the integer as an argument, and return `True` if the integer is prime, otherwise it must return `False`.

Call your function with the argument as `197`.

(4 points)

C.2.2 Factor

Write a function that checks if an integer is a factor of another integer. The function must accept both the integers as arguments, and return `True` if the integer is a factor, otherwise it must return `False`.

Call your function with the arguments as `(19,85)`.

(3 points)

C.2.3 Prime Factors

Prompt the user to input a positive integer. Use the functions in B.2.1 and B.2.2 to print the prime factors of the integer. Your program should be no more than 4 lines (excluding the comments)

Check your program is the user inputs 190

(8 points)

C.3 Binary search

C.3.1 Word search

The tuple below named as `tuple_of_words` consists of words. Write a function that accepts a word, say `word_to_search` and the `tuple_of_words` as arguments, and finds if the `word_to_search` occurs in the `tuple_of_words` or not. This is very simple to do with the code `word_to_search in tuple_of_words`. However, this code is unfortunately very slow.

As the words in the `tuple_of_words` are already sorted in alphabetical order, we can search using a faster way, called binary search. To implement binary search in a function, start by comparing `word_to_search` with the middle entry in the `tuple_of_words`. If they are equal, then you are done and the function should return `True`. On the other hand, if the `word_to_search` comes before the middle entry, then search the first half of `tuple_of_words`. If it comes after the middle entry, then search the second half of `tuple_of_words`. Then repeat the process on the appropriate half of the `tuple_of_words` and continue until the word is found or there is nothing left to search, in which case the function short return `False`. The `<` and `>` operators can be used to alphabetically compare two strings.

You may write just one function or multiple functions to solve this problem.

Check your function if the `word_to_search` is:

1. `'rocket'`

2. `'rest'`
3. `'ambush'`

(25 points)

```
tuple_of_words=('abacus', 'abdomen', 'abdominal', 'abide', 'abiding', 'ability', 'ablaze',
                'cattishly', 'cattle', 'catty', 'catwalk', 'caucasian', 'caucus', 'causal'
    'directly', 'directory', 'direness', 'dirtiness', 'disabled', 'disagree', 'disallow',
    'freemason', 'freeness', 'freestyle', 'freeware', 'freeway', 'freewill', 'freezable',
    'laurel', 'lavender', 'lavish', 'laxative', 'lazily', 'laziness', 'lazy', 'lecturer',
    'payee', 'payer', 'paying', 'payment', 'payphone', 'payroll', 'pebble', 'pebbly', 'pec
    'rift', 'rigging', 'rigid', 'rigor', 'rimless', 'rimmed', 'rind', 'rink', 'rinse', 'ri
    'stoneware', 'stonework', 'stoning', 'stony', 'stood', 'stooge', 'stool', 'stoop', 'st
    'unscented', 'unscrew', 'unsealed', 'unseated', 'unsecured', 'unseeing', 'unseemly', '
```

C.3.2 Iterations to find the word

Update the function in B.3.1 to also print the number of iterations it took to find the `word_to_search` or fail in finding the `word_to_search`.

Check your function if the `word_to_search` is:

1. `'rocket'`
2. `'rest'`
3. `'amendable'`

(10 points)

C.3.3 Index of word

Update the function in B.3.2 to also print the index of `word_to_search` in `tuple_of_words` if the word is found in the tuple. For example, the index of 'abacus' is 0, the index of 'abdomen' is 1, and so on.

Check your function if the `'word_to_search'` is:

1. `'rocket'`
2. `'rest'`
3. `'ambush'`

(10 points)

C.3.4 Maximum iterations

What is the maximum number of iterations it may take for your function to search or fail in searching the `word_to_search`. You may either write a program to answer this question, or answer it analytically.

(5 points)

D Assignment C

Instructions

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. There are 5 points for clealiness and organization. The breakdow is as follows:

- Must be an HTML file rendered using Quarto (1.5 pts).
- There aren't excessively long outputs of extraneous information (e.g. no printouts of unnecessary results without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)
- The code should be commented and clearly written with intuitive variable names. For example, use variable names such as number_input, factor, hours, instead of a,b,xyz, etc. (1.5 pts)

6. The assignment is worth 100 points, and is due on **29th April 2023 at 11:59 pm**.

D.1 GDP of The USA

USA's GDP per capita from 1960 to 2021 is given by the tuple `T` in the code cell below. The values are arranged in ascending order of the year, i.e., the first value is for 1960, the second value is for 1961, and so on.

```
T = (3007, 3067, 3244, 3375,3574, 3828, 4146, 4336, 4696, 5032,5234,5609,6094,6726,7226,78
```

D.1.1 Gaps

Use list comprehension to produce a list of the gaps between consecutive entries in `T`, i.e, the increase in GDP per capita with respect to the previous year. The list with gaps should look like: [60, 177, ...].

(6 points)

D.1.2 Maximum gap size

Use the list developed in C.1.1 to find the maximum gap size, i.e, the maximum increase in GDP per capita.

(2 points)

D.1.3 Gaps higher than $1000

Using list comprehension with the list developed in C.1.1, find the percentage of gaps that have size greater than $1000.

(6 points)

D.1.4 Dictionary

Create a dictionary `D`, where the `key` is the year, and `value` for the `key` is the increase in GDP per capita in that year with respect to the previous year, i.e., the gaps computed in C.1.1.

(6 points)

D.1.5 Maximum increase

Use the dictionary `D` to find the year when the GDP per capita increase was the maximum as compared to the previous year. Use the list comprehension method.

(6 points)

Hint: [..... for .... in D.items() if .....]

D.1.6 GDP per capita decrease

Use the dictionary `D` to find the years when the GDP per capita decreased with respect to the previous year. Use the list comprehension method.

(6 points)

D.2 Ted Talks

D.2.1 Reading data

Read the file *TED_Talks.json* on ted talks using the code below. You will get the data in the object `TED_Talks_data`. Just look at the data structure of `TED_Talks_data`. You will need to know how the data is structured in lists/dictionaries to answer the questions below.

Note that the data must be stored in the same directory as the notebook.

(2 points)

```
import json
with open("TED_Talks.json", "r") as file:
    TED_Talks_data=json.load(file)
```

D.2.2 Number of talks

Find the number of talks in the dataset.

(2 points)

D.2.3 Popular talk

Find the `headline`, `speaker` and `year_filmed` of the talk with the highest number of `views`.

(6 points)

D.2.4 Mean and median views

What are the mean and median number of `views` for a talk? Can we say that the majority of talks (i.e., more than 50% of the talks) have less `views` than the average number of `views` for a talk? Justify your answer.

(6 points)

D.2.5 Views vs average views

Do at least 25% of the talks have more `views` than the average number of `views` for a talk? Justify your answer.

(4 points)

D.2.6 `Confusing` talks

Find the `headline` of the talk that received the highest number of votes in the `Confusing` category.

(8 points)

D.2.7 `Fascinating` talks

Find the `headline` and the `year_filmed` of the talk that received the highest percentage of votes in the *Fascinating* category.

$$\text{Percentage of } \textit{Fascinating} \text{ votes for a ted talk} = \frac{Number\ of\ votes\ in\ the\ Fascinating\ category}{Total\ votes\ in\ all\ categories}$$

(10 points)

D.3 Poker

The object `deck` defined below corresponds to a deck of cards. Estimate the probability that a five card hand will be:

1. Straight
2. Three-of-a-kind
3. Two-pair
4. One-pair
5. High card

You may check the meaning of the above terms here.

(25 points)

Hint:

Estimate these probabilities as follows.

1. Write a function that accepts a hand of 5 cards as argument, and returns relevant characterisitics of a hand, such as the number of distinct card values, maximum occurences of a value etc. Using the values returned by this function (may be in a dictionary), you can compute if the hand is of any of the above types *(Straight / Three-of-a-kind / two-pair / one-pair / high card)*.
2. Randomly pull a hand of 5 cards from the `deck`. Call the function developed in (1) to get the relevant characteristics of the hand. Use those characteristics to determine if the hand is one of the five mentioned types *(Straight / Three-of-a-kind / two-pair / one-pair / high card)*.
3. Repeat (2) 10,000 times.
4. Estimate the probability of the hand being of the above five mentioned types *(Straight / Three-of-a-kind / two-pair / one-pair / high card)* from the results of the 10,000 simulations.

You may use the function `shuffle()` from the library `random` to shuffle the deck everytime before pulling a hand of 5 cards.

You don't need to stick to the hint if you feel you have a better way to do it. In case you have a better way, you can claim 10 bonus points for this assignment.

```
deck = [{'value':i, 'suit':c}
for c in ['spades', 'clubs', 'hearts', 'diamonds']
```

```
for i in range(2,15)]
```

E Assignment templates and Datasets

Assignment templates and datasets used in the book can be found here