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 R: Recursion

1.1 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
```

1.1.1 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)
```

1.1.2 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)
```

1.1.3 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)
```

1.2 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