

# The 100-page ChatGPT Generated Python Tutorial

For Absolute Beginners

Created by Vladimir Ilievski x ChatGPT  
iSquared.digital



## Exercise 1

Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5, between 2000 and 3200 (both included). The numbers obtained should be printed in a comma-separated sequence on a single line.

### Code

```
for i in range(2000, 3201):
    if i % 7 == 0 and i % 5 != 0:
        print(i, end=",")
print("\b")
```

### ChatGPT Explanation

This code snippet outputs all the numbers between 2000 and 3200 (inclusive) that are divisible by 7 but not divisible by 5.

The `for` loop iterates over the range of numbers from 2000 to 3200 and the `if` statement inside the loop checks if the current number `i` is divisible by 7 but not divisible by 5. If the condition is true, then the current number `i` is printed, followed by a comma.

Finally, the `\b` at the end of the code removes the last comma.

## Exercise 2

Write a program which can compute the factorial of a given numbers. The results should be printed in a comma-separated sequence on a single line. Suppose the following input is supplied to the program: 8 Then, the output should be:40320

### Code

```
n = int(input())
fact = 1
for i in range(1, n + 1):
    fact = fact * i
print(fact)
```

### ChatGPT Explanation

This code snippet is a simple program that calculates the factorial of a given number.

The first line of code is getting an input from the user and assigns it to the variable `n`. The `int` function is used to convert the input to an integer.

The next line creates a variable called `fact` and assigns it the value of 1.

The following `for` loop starts with the variable `i` set to 1, and it will run as long as `i` is less than or equal to `n + 1`, and `i` will increment by 1 after each iteration.

In each iteration, the value of `fact` is multiplied by `i` and the result is stored in `fact` again, so on each iteration `fact` will be updated with the new value.

After the loop has finished, the final value of `fact` is printed using the `print()` function.

## Exercise 3

With a given integral number  $n$ , write a program to generate a dictionary that contains  $(i, i \times i)$  such that  $i$  is an integral number between 1 and  $n$  (both included). and then the program should print the dictionary. Suppose the following input is supplied to the program: 8

### Code

```
n = int(input())
ans = {}
for i in range(1, n + 1):
    ans[i] = i * i
print(ans)
```

### ChatGPT Explanation

This code snippet is a simple program that creates a dictionary where the keys are the integers from 1 to  $n$ , and the values are the square of the corresponding key.

The first line of code is getting an input from the user and assigns it to the variable `n`. The `int` function is used to convert the input to an integer.

The next line creates an empty dictionary called `ans`.

The following for loop starts with the variable `i` set to 1, and it will run as long as `i` is less than or equal to `n + 1`, and `i` will increment by 1 after each iteration.

In each iteration, the `i` is used as key in the dictionary `ans` and the value is calculated as `i * i` and assigned to that key. After the loop has finished, the final dictionary `ans` is printed using the `print()` function.

## Exercise 4

Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple which contains every number. Suppose the following input is supplied to the program:

### Code

```
lst = input().split(",")
tpl = tuple(lst)

print(lst)
print(tpl)
```

### ChatGPT Explanation

This code snippet is a simple program that takes a string of comma-separated values as input, converts it to a list and then converts it to a tuple.

The first line of code is getting an input from the user and assigns it to the variable `lst`. The `split()` method is used to convert the input string into a list of strings by splitting it at each comma.

The next line of code creates a variable called `tpl`, and it uses the `tuple()` function to convert the `lst` variable from a list to a tuple.

The last two lines of code are using the `print()` function to display the `lst` variable and the `tpl` variable on the screen.

## Exercise 5

Define a class which has at least two methods: - `getString`: to get a string from console input - `printString`: to print the string in upper case.

### Code

```
class IOstring:
    def __init__(self):
        pass

    def get_string(self):
        self.s = input()

    def print_string(self):
        print(self.s.upper())
```

```
xx = IOstring()
xx.get_string()
xx.print_string()
```

### ChatGPT Explanation

This code snippet is a simple program that defines a class called `IOstring`, which has two methods: `get_string()` and `print_string()`.

The `__init__` method is the constructor of the class, it is called when an object of this class is created and does not have any specific functionality in this case, it just pass.

The `get_string()` method prompts the user to input a string and assigns it to the variable `s` as an attribute of the class.

The `print_string()` method then takes the value stored in the `s` attribute, converts it to uppercase using the `upper()` method, and prints it out.

The last three lines of code are creating an object of the `IOstring` class called `xx`, calling the `get_string()` method on it to set the value of the `s` attribute, and then calling the `print_string()` method to print the value of `s` in upper-case.

## Exercise 6

Write a program that calculates and prints the value according to the given formula:

$$Q = \text{Square root of } [(2 * C * D) / H]$$

Following are the fixed values of C and H:

- C is 50. H is 30.

D is the variable whose values should be input to your program in a comma-separated sequence. For example Let us assume the following comma separated input sequence is given to the program:

### Code

```
from math import sqrt

C, H = 50, 30

def calc(D):
    return sqrt((2 * C * D) / H)

print(",".join([str(int(calc(int(i)))) for i in input().split(",")]))
```

### ChatGPT Explanation

This code is a simple program that takes a string of comma-separated values as input, performs some calculations on each of the values, and then prints the result as a string of comma-separated values.

The first line of code imports the `sqrt()` function from the `math` module, which is used to calculate the square root of a number.

The next two lines define two constants, `C` and `H`, which are used in the calculations.

The following function `calc(D)` takes a parameter `D` and returns the result of the calculation `sqrt((2 * C * D) / H)`

The final line of code uses the `input()` function to get user input, which is expected to be a string of comma-separated values. It then uses the `split()` method to convert the input string into a list of strings by splitting it at each comma.

It then creates a list comprehension that iterates over the list of input values, calls the `calc()` function on each input value, which returns a float, and then converts the float to an int using the `int()` function. Then it uses the `join()` method to join the list of integers into a string of comma-separated values and prints it out.

## Exercise 7

Write a program which takes 2 digits, X, Y as input and generates a 2-dimensional array. The element value in the i-th row and j-th column of the array should be i.j.

### Code

```
x, y = map(int, input().split(","))
lst = [[i * j for j in range(y)] for i in range(x)]
print(lst)
```

### ChatGPT Explanation

This code is a simple program that takes two integers as input, separated by a comma and creates a list of lists (also known as a 2D list) by using a nested list comprehension.

The first line of code uses the `input()` function to get user input, which is expected to be two integers separated by a comma. It then uses the `split()` method to separate the input string at the comma and the `map()` function to convert the resulting two strings into integers and assigns them to the variables `x`, `y`.

The second line of code creates a list of lists called `lst` using a nested list comprehension. The outer list comprehension iterates over the range of `x`, and the inner list comprehension iterates over the range of `y`, and it creates a new list of integers where each integer is the product of the outer loop variable and the inner loop variable.

The last line of code prints out the list of lists.



## Exercise 8

Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.

### Code

```
lst = input().split(",")
lst.sort()
print(",".join(lst))
```

### ChatGPT Explanation

This code is a simple program that takes a string of comma-separated values as input, sorts the values in alphabetical or numerical order, and then prints the sorted values as a string of comma-separated values.

The first line of code uses the `input()` function to get user input, which is expected to be a string of comma-separated values, and the `split()` method to convert the input string into a list of strings by splitting it at each comma.

The second line of code sorts the list of strings using the `sort()` method. This sorts the list in alphabetical order for strings and numerical order for numbers.

The final line of code uses the `join()` method to join the list of strings into a string of comma-separated values, and then prints it out.

## Exercise 9

Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized.

### Code

```
def user_input():
    while True:
        s = input()
        if not s:
            return
        yield s

for line in map(str.upper, user_input()):
    print(line)
```

### ChatGPT Explanation

This code defines a function `user_input()` that takes no arguments and continuously prompts the user for input until they provide an empty string (by just pressing enter). Each input string provided by the user is then passed to the `str.upper()` method to convert the string to uppercase and then passed to the `print()` function to print it out. The `map()` function applies `str.upper()` method to each input string.

The `while` loop in the function `user_input()` uses the `input()` function to continuously prompt the user for input and assigns the input string to the variable `s`.

The `if not s:` line checks whether the input string is empty or not. If it is empty, the `return` statement is executed, which terminates the function and the loop stops. If the input string is not empty, it is passed to the `yield` statement, which returns the input string to the calling code and continues to execute the next iteration of the loop.

The `for` loop uses the `user_input()` function to get user input and the `map()` function to call the `str.upper()` method on each input string to convert it to uppercase and the `print()` function to print it out.

## Exercise 10

Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically.

### Code

```
word = sorted(  
    list(set(input().split()))  
)  
print(" ".join(word))
```

### ChatGPT Explanation

This code is a simple program that takes a string of words as input, sorts them in alphabetical order and removes any duplicates, and then prints out the unique and sorted words as a string.

The first line of code uses the `input()` function to get user input, which is expected to be a string of words, and the `split()` method to convert the input string into a list of words by splitting it at each space.

Then it converts the list of words into a set with the `set()` method, which removes any duplicates. Then the set is converted back to a list with `list()` method.

The second line sorts the list of words using the `sorted()` method. This sorts the list in alphabetical order.

The final line of code uses the `join()` method to join the list of words into a string of words separated by space, and then prints it out.

## Exercise 11

Write a program which accepts a sequence of comma separated 4 digit binary numbers as its input and then check whether they are divisible by 5 or not. The numbers that are divisible by 5 are to be printed in a comma separated sequence.

### Code

```
data = input().split(",")
data = list(
    filter(lambda i: int(i, 2) % 5 == 0, data)
) # lambda is an operator that helps to write function of one line
print(",".join(data))
```

### ChatGPT Explanation

This code takes a string of comma-separated binary numbers as input, filters out the numbers that are not divisible by 5 when converted to decimal, and then prints the remaining numbers as a comma-separated string.

The first line of code uses the `input()` function to get user input, which is expected to be a string of comma-separated binary numbers. It then uses the `split()` method to convert the input string into a list of binary numbers by splitting it at each comma.

The next line of code uses the `filter()` function to filter out the binary numbers that are not divisible by 5 when converted to decimal. `filter()` function applies the lambda function to each element in the list, the lambda function takes one argument `i`, converts it to decimal using `int(i, 2)` and check whether it's divisible by 5 or not, if it's true it keeps the element otherwise it filters out the element.

Finally, the `join()` method is used to join the filtered list of binary numbers into a string of comma-separated binary numbers and then the string is printed out.

## Exercise 12

Write a program, which will find all such numbers between 1000 and 3000 (both included) such that each digit of the number is an even number. The numbers obtained should be printed in a comma-separated sequence on a single line.

### Code

```
lst = [str(i) for i in range(1000, 3001)]
lst = list(
    filter(lambda i: all(ord(j) % 2 == 0 for j in i), lst)
) # using lambda to define function inside filter function
print(",".join(lst))
```

### ChatGPT Explanation

This code generates a list of numbers between 1000 and 3000 and filters out the numbers that have any odd-numbered digits in them.

The first line of code uses a list comprehension to generate a list of numbers between 1000 and 3000 and converts them to strings.

The next line of code uses the `filter()` function to filter out the numbers that have any odd-numbered digits in them. `filter()` function applies the lambda function to each element in the list, the lambda function takes one argument `i`, and for each digit `j` in `i`, it checks whether the ASCII value of `j` is even using the `ord()` function and the modulus operator `%`. If all digits are even, the lambda function returns `True`, and the number is kept in the list, otherwise it filters out the number.

Finally, the `join()` method is used to join the filtered list of numbers into a string of comma-separated numbers and then the string is printed out.

## Exercise 13

Write a program that accepts a sentence and calculate the number of letters and digits.

### Code

```
word = input()
letter, digit = 0, 0

for i in word:
    if i.isalpha(): # returns True if alphabet
        letter += 1
    elif i.isnumeric(): # returns True if numeric
        digit += 1
print(
    f"LETTERS {letter}\n{digit}"
) # two different types of formating method is shown in both solution
```

### ChatGPT Explanation

This code takes an input string, and then it iterates over each character in the string.

For each character, it checks if the character is an alphabet using the `isalpha()` method. If it is an alphabet, the variable `letter` is incremented by 1.

Then, it checks if the character is a digit using the `isnumeric()` method. If it is a digit, the variable `digit` is incremented by 1.

Finally, it prints the count of letters and digits in the input string. `f"LETTERS {letter}\n{digit}"` is used to format the output string where `letter` and `digit` are the variables that contain the count of letters and digits respectively in the input word.

## Exercise 14

Write a program that accepts a sentence and calculate the number of upper case letters and lower case letters.

### Code

```
string = input("Enter the sentence")
upper = 0
lower = 0
for x in string:
    if x.isupper() == True:
        upper += 1
    if x.islower() == True:
        lower += 1

print("UPPER CASE: ", upper)
print("LOWER CASE: ", lower)
```

### ChatGPT Explanation

This code takes an input string, and then it iterates over each character in the string.

For each character, it checks if the character is uppercase using the `isupper()` method. If it is uppercase, the variable `upper` is incremented by 1.

Then, it checks if the character is lowercase using the `islower()` method. If it is lowercase, the variable `lower` is incremented by 1.

Finally, it prints the count of uppercase and lowercase characters in the input string.

## Exercise 15

Write a program that computes the value of  $a+aa+aaa+aaaa$  with a given digit as the value of `a`.

### Code

```
a = input()
total, tmp = 0, str() # initialing an integer and empty string

for i in range(4):
    tmp += a # concatenating 'a' to 'tmp'
    total += int(tmp) # converting string type to integer type

print(total)
```

### ChatGPT Explanation

This code takes an input, and then assigns it to a variable `a`.

It then initializes two variables, `total` and `tmp`, with the values of 0 and empty string respectively.

The code then uses a `for` loop that iterates 4 times. In each iteration, it concatenates the variable `a` to the variable `tmp` and then converts it to an integer and adds it to the variable `total`.

At the end, it prints the total.



## Exercise 16

Use a list comprehension to square each odd number in a list. The list is input by a sequence of comma-separated numbers. Suppose the following input is supplied to the program:

### Code

```
lst = [str(int(i) ** 2) for i in input().split(",") if int(i) % 2]
print(",".join(lst))
```

### ChatGPT Explanation

This code takes an input, which is expected to be a comma-separated list of numbers, converts it into a list of strings, then it applies a list comprehension on it.

The list comprehension iterates over each element of the input list, checks if it is odd by checking if the integer representation of the element modulo 2 is non-zero. If it is odd, it takes the square of the element and converts it back to a string before appending it to the new list.

At the end, it uses the `join` method to join the elements of the new list with commas and prints it.

## Exercise 17

Write a program that computes the net amount of a bank account based a transaction log from console input. The transaction log format is shown as following:

```
D 100
W 200
```

D means deposit while W means withdrawal.

### Code

```
lines = []
while True:
    loopInput = input()
    if loopInput == "done":
        break
    else:
        lines.append(loopInput)

lst = list(int(i[2:]) if i[0] == "D" else -int(i[2:]) for i in lines)
print(sum(lst))
```

### ChatGPT Explanation

This code snippet first creates an empty list called `lines`.

It then enters a `while` loop that continues until the user inputs "done". For each iteration of the loop, the script takes in user input and checks if it is "done". If it is not, it appends the input to the `lines` list.

Once the loop breaks, the script creates a new list called `lst` using list comprehension. It iterates through each element of `lines` and checks if the first character of the element is "D". If it is, it adds the element as is to the new list, otherwise it adds the element as a negative number.

Finally, the script calculates the sum of the elements in the `lst` list and prints it.

## Exercise 18

A website requires the users to input username and password to register. Write a program to check the validity of password input by users. Following are the criteria for checking the password:

- At least 1 letter between [a-z]
- At least 1 number between [0-9]
- At least 1 letter between [A-Z]
- At least 1 character from [!#\$%&@]
- Minimum length of transaction password: 6
- Maximum length of transaction password: 12

Your program should accept a sequence of comma separated passwords and will check them according to the above criteria. Passwords that match the criteria are to be printed, each separated by a comma.

### Code

```
def is_low(x): # Returns True if the string has a lowercase
    for i in x:
        if "a" <= i and i <= "z":
            return True
    return False

def is_up(x): # Returns True if the string has a uppercase
    for i in x:
        if "A" <= i and i <= "Z":
            return True
    return False

def is_num(x): # Returns True if the string has a numeric digit
    for i in x:
        if "0" <= i and i <= "9":
            return True
    return False

def is_other(x): # Returns True if the string has any "!#$%&@"
    for i in x:
        if i == "!" or i == "#" or i == "@":
            return True
    return False

s = input().split(",")
lst = []

for i in s:
```

```

length = len(i)
if (
    6 <= length
    and length <= 12
    and is_low(i)
    and is_up(i)
    and is_num(i)
    and is_other(i)
): # Checks if all the requirements are fulfilled
    lst.append(i)

print(",".join(lst))

```

## ChatGPT Explanation

This script takes input of a string of comma-separated values, and checks each value against certain conditions. The conditions are:

1. The length of the string is between 6 and 12 characters
2. The string contains at least one lowercase letter
3. The string contains at least one uppercase letter
4. The string contains at least one numeric digit
5. The string contains at least one of the special characters "\$#@".

The script uses 4 separate functions (`is_low`, `is_up`, `is_num`, `is_other`) to check if each string meets each of these requirements.

If a value meets all these conditions, it is added to a list. Finally, the script prints all the values in the list, joined by commas.

The function `is_low(x)` checks if the input string `x` has any lowercase letters by iterating through each character in the string and checking if it falls within the range of lowercase letters. It returns `True` if at least one lowercase letter is found, and `False` otherwise.

The function `is_up(x)` returns `True` if the input string `x` contains at least one uppercase letter, and `False` otherwise. It iterates through each character in the input string, and checks if it is within the range of uppercase letters using the comparison operator `<=`. If it finds an uppercase letter, it immediately returns `True`. If it reaches the end of the input string without finding an uppercase letter, it returns `False`.

The function `is_num(x)` takes in a single argument `x` which is a string. The function then uses a `for` loop to iterate through each character in the string `x`. Within the `for` loop, there is an `if` statement that checks if the current character being iterated through is a numeric digit (i.e. between 0 and 9). If the current character is a numeric digit, the function immediately returns `True`. If the `for` loop completes without finding a numeric digit, the function returns `False` at

the end. In other words, this function checks if the input string contains any numeric digits, and returns `True` if it does and `False` if it does not.

The function `is_other(x)` takes in a single input, `x`. The function then iterates through each character in the input `x` using a for loop. Inside the loop, the function checks if the current character is equal to any of the special characters `$`, `#`, or `@`. If any of these characters are present in the input, the function immediately returns `True`. If the loop completes without finding any of these characters, the function returns `False` indicating that none of the special characters were found in the input.

## Exercise 19

You are required to write a program to sort the (name, age, score) tuples by ascending order where name is string, age and score are numbers. The tuples are input by console. The sort criteria is:

1. Sort based on name
2. Then sort based on age
3. Then sort by score

### Code

```
lst = []
while True:
    s = input().split(",")
    if not s[0]: # breaks for blank input
        break
    lst.append(tuple(s))

lst.sort(
    key=lambda x: (x[0], x[1], x[2])
) # here key is defined by lambda and the data is sorted by element priority 0>1>2 in ascending order
print(lst)
```

### ChatGPT Explanation

This code snippet is a Python program that prompts the user for input of strings separated by commas.

It creates an empty list called `lst` and enters a `while` loop. Within the `while` loop, it takes user input, splits it into a list of strings by the commas, and assigns it to the variable `s`. If the first element of `s` is blank, the loop breaks. Otherwise, it appends a tuple of the elements of `s` to `lst`.

After the loop, the program sorts `lst` by the first, second, and third elements of each tuple, and then prints `lst`.

## Exercise 20

Define a class with a generator which can iterate the numbers, which are divisible by 7, between a given range 0 and n.

### Code

```
class MyGen:
    def by_seven(self, n):
        for i in range(0, int(n / 7) + 1):
            yield i * 7

for i in MyGen().by_seven(int(input("Please enter a number... "))):
    print(i)
```

### ChatGPT Explanation

This code defines a class called `MyGen` that has a method called `by_seven`. The method takes in a number `n` as an argument and uses a `for` loop to iterate over a range of numbers that are multiples of 7 and less than or equal to `n`. The method uses the `yield` keyword to return each multiple of 7 one at a time as the `for` loop iterates through the range.

After the class is defined, the code prompts the user to enter a number, and then it creates an instance of the `MyGen` class and calls the `by_seven` method with the user-provided number as an argument. The code then uses a `for` loop to print out each number generated by the `by_seven` method one at a time.

## Exercise 21

A robot moves in a plane starting from the original point (0,0). The robot can move toward UP, DOWN, LEFT and RIGHT with a given steps. The trace of robot movement is shown as the following:

- UP 5
- DOWN 3
- LEFT 3
- RIGHT 2

The numbers after the direction are steps. Please write a program to compute the distance from current position after a sequence of movement and original point. If the distance is a float, then just print the nearest integer. Example: If the following tuples are given as input to the program:

### Code

```
import math

x, y = 0, 0
while True:
    s = input().split()
    if not s:
        break
    if s[0] == "UP": # s[0] indicates command
        x -= int(s[1]) # s[1] indicates unit of move
    if s[0] == "DOWN":
        x += int(s[1])
    if s[0] == "LEFT":
        y -= int(s[1])
    if s[0] == "RIGHT":
        y += int(s[1])
        # N**P means N^P
dist = round(
    math.sqrt(x ** 2 + y ** 2)
) # euclidean distance = square root of (x^2+y^2) and rounding it to nearest integer
print(dist)
```

### ChatGPT Explanation

This code snippet is a Python program that calculates the distance from the origin (0,0) to a point on the x-y plane.

The program takes input as a series of commands (UP, DOWN, LEFT, RIGHT) and an integer value indicating the number of units to move in that direction. The input is repeatedly taken in a `while` loop until a blank input is given.



The program keeps track of the  $x$  and  $y$  coordinates as the input commands are executed and then calculates the distance from the origin to the final point using the Pythagorean theorem (the square root of  $x^2 + y^2$ ). The calculated distance is rounded off and then printed.

## Exercise 22

Write a program to compute the frequency of the words from the input. The output should output after sorting the key alphanumerically.

### Code

```
from pprint import pprint

p = input().split()
pprint({i: p.count(i) for i in p})
```

### ChatGPT Explanation

This code snippet in Python is used for counting the occurrences of each element in a given list and printing the results in a dictionary format.

The input is taken as a string of elements separated by a space, which is then split into a list of elements using the `split` method. The `pprint` module is imported to format the output as a dictionary. A dictionary comprehension is used to iterate over the list of elements and count the occurrences of each element. The resulting dictionary is then printed using the `pprint` function.

## Exercise 23

Write a method which can calculate square value of number

### Code

```
n = int(input())  
print(n ** 2)
```

### ChatGPT Explanation

This code snippet prompts the user to input an integer, and then raises that integer to the power of 2 ( $n^2$ ) and prints the result. For example, if the user inputs the number 5, the program would calculate  $5^2$  and print 25.

## Exercise 24

Please write a program to print some Python built-in functions documents, such as `abs()`, `int()`, `raw_input()`. And add document for your own function.

### Code

```
def pow(n, p):  
    """  
    param n: This is any integer number  
    param p: This is power over n  
    return: n to the power p = n^p  
    """  
    return n ** p  
  
print(pow(3, 4))  
print(pow.__doc__)
```

### ChatGPT Explanation

This is a Python function definition for a function called `pow` that takes in two parameters, `n` and `p`.

The function calculates and returns the result of raising the value of `n` to the power of `p` ( $n$  to the power of  $p = n^p$ ). The function also has a `docstring` that describes the purpose and parameters of the function.

The last two lines of the code snippet are calling the `pow` function with the values 3 and 4 as arguments, which will return  $3^4 = 81$  and also prints the `docstring` of the function.

## Exercise 25

Define a class, which have a class parameter and have a same instance parameter.

### Code

```
class Car:
    name = "Car"
    def __init__(self, name=None):
        self.name = name

honda = Car("Honda")
print(f"{Car.name} name is {honda.name}")

toyota = Car()
toyota.name = "Toyota"
print(f"{Car.name} name is {toyota.name}")
```

### ChatGPT Explanation

This code defines a class called `Car` with a class variable `name` which is set to "Car" by default. When creating a new object of the `Car` class, you can pass in a custom name as an argument, otherwise it will use the default "Car" value.

The first instance of the class is created and assigned to the variable `honda`. The class variable `name` is overridden with the value "Honda" that was passed in during the object's creation. Then a message is printed, displaying the class variable's value of "Car" and the object variable's value of "Honda".

The second instance of the class is created and assigned to the variable `toyota`. In this case, the name of the object is not passed during the object creation, so the default value of "Car" is used. Later, the object variable's value is overridden with "Toyota" and a message is printed, displaying the class variable's value of "Car" and the object variable's value of "Toyota".

## Exercise 26

Define a function which can compute the sum of two numbers.

### Code

```
sum = lambda n1, n2: n1 + n2 # here lambda is use to define little function as sum  
print(sum(1, 2))
```

### ChatGPT Explanation

This code creates an anonymous function called `sum` that takes in two parameters, `n1` and `n2`, and returns their sum. The function is then immediately called with the arguments 1 and 2, and the result of that function call (3) is printed.

## Exercise 27

Define a function that can convert an integer into a string and print it in console.

### Code

```
conv = lambda x: str(x)
n = conv(10)
print(n)
print(type(n)) # checks the type of the variable
```

### ChatGPT Explanation

This code defines a lambda function named `conv`, which takes in a single argument `x` and returns the string representation of that argument.

It then assigns the value of the function applied to the integer 10 to a variable `n`. It then prints out the value of `n` and its data type using the `print` and `type` function respectively. It will output:

```
10
<class 'str'>
```

## Exercise 28

Define a function that can receive two integer numbers in string form and compute their sum and then print it in console.

### Code

```
sum = lambda s1, s2: int(s1) + int(s2)
print(sum("10", "45")) # 55
```

### ChatGPT Explanation

This code defines a lambda function called `sum`, which takes in two parameters `s1` and `s2` and converts them from strings to integers using the `int()` function.

Then it performs the sum of the integers and returns the result. Then, the code calls the `sum` function with the arguments "10" and "45" and prints the result, which is 55.



## Exercise 29

Define a function that can accept two strings as input and concatenate them and then print it in console.

### Code

```
sum = lambda s1, s2: s1 + s2
print(sum("10", "45")) # 1045
```

### ChatGPT Explanation

This code defines a lambda function named `sum` that takes in two arguments, `s1` and `s2`, and returns their concatenation as a string. Then it calls this lambda function with the arguments `"10"` and `"45"` and prints the result, which is `"1045"`.

## Exercise 30

Define a function that can accept two strings as input and print the string with maximum length in console. If two strings have the same length, then the function should print all strings line by line.

### Code

```
func = (  
    lambda a, b: print(max((a, b), key=len))  
    if len(a) != len(b)  
    else print(a + "\n" + b)  
)
```

### ChatGPT Explanation

This code defines a lambda function `func` that takes in two arguments `a` and `b`.

The function then checks if the lengths of `a` and `b` are not equal. If they are not, it prints the argument with the larger length using the `max` function and the `len` key. If the lengths are equal, it prints both `a` and `b` on separate lines using string concatenation. Note that this function definition doesn't execute it.

You would need to call the function by passing two parameters.

## Exercise 31

Define a function which can print a dictionary where the keys are numbers between 1 and 20 (both included) and the values are square of keys.

### Code

```
def printDict():  
    dict = {i: i ** 2 for i in range(1, 21)}  
    print(dict)  
  
printDict()
```

### ChatGPT Explanation

This code defines a function `printDict()`. When called, the function creates a dictionary, `dict`, with keys being numbers from 1 to 20, and values being the square of the corresponding key. It then prints the dictionary to the console.

## Exercise 32

Define a function which can generate a dictionary where the keys are numbers between 1 and 20 (both included) and the values are square of keys. The function should just print the keys only.

### Code

```
def printDict():
    dict = {i: i ** 2 for i in range(1, 21)}
    print(dict.keys()) # print keys of a dictionary

printDict()
```

### ChatGPT Explanation

This code defines a function called `printDict()`. Inside the function, it creates a dictionary called `dict` with keys being integers from 1 to 20, and values being the square of the corresponding key.

Then it calls the `keys()` method on the `dict` object, which returns a view object of all the keys in the dictionary, and prints it. When the function is called, it will print all the keys in the dictionary (1, 2, 3, 4, ..., 20) without values.

## Exercise 33

Define a function which can generate and print a list where the values are square of numbers between 1 and 20 (both included).

### Code

```
def printList():  
    lst = [i ** 2 for i in range(1, 21)]  
    print(lst)  
  
printList()
```

### ChatGPT Explanation

This code defines a function called `printList()` that creates a list of the squares of the numbers from 1 to 20 (inclusive). It then prints the list to the console.

When the function is called (by the last line `printList()`), it will execute the instructions inside it and print the list of the squares of the numbers from 1 to 20 to the console.

## Exercise 34

Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print the first 5 elements in the list.

### Code

```
func = lambda: ([i ** 2 for i in range(1, 21)][:5])
print(*(func()), sep="\n")
```

### ChatGPT Explanation

This code defines a lambda function `func` that returns a list of the first 5 squares of the integers from 1 to 20.

The lambda function is immediately called and the result is unpacked and passed as separate arguments to the `print` function, which then separates the items in the list with newlines and prints them. The output will be a list of the first five square of numbers 1 to 20.

## Exercise 35

Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print the last 5 elements in the list.

### Code

```
def printList():
    lst = [i ** 2 for i in range(1, 21)]
    for i in range(19, 14, -1):
        print(lst[i])

printList()
```

### ChatGPT Explanation

This function creates a list of the squares of the numbers from 1 to 20 using a list comprehension. Then, it uses a `for` loop to iterate through the list in reverse order, starting from the 19th index and going to the 14th index (with a step of -1, to count backwards). For each iteration, the function prints the value of the corresponding element in the list. So it will print the squares of the numbers 15 to 20 in descending order.

## Exercise 36

Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print all values except the first 5 elements in the list.

### Code

```
def printList():
    lst = [i ** 2 for i in range(1, 21)]
    for i in range(5, 20):
        print(lst[i])

printList()
```

### ChatGPT Explanation

This code defines a function called `printList()`. Inside the function, it creates a list called `lst` that contains the squares of the numbers from 1 to 20. Then it uses a `for` loop to iterate through a range of numbers from 5 to 20 (not including 20), and for each iteration, it prints the element of the `lst` list at that index.

So it will print the elements of the list that are at index 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, and 19. Essentially, it is printing the last 15 elements of the list.



## Exercise 37

Define a function which can generate and print a tuple where the value are square of numbers between 1 and 20 (both included).

### Code

```
def square_of_numbers():  
    return tuple(i ** 2 for i in range(1, 21))  
  
print(square_of_numbers())
```

### ChatGPT Explanation

This code defines a function `square_of_numbers()` that creates and returns a tuple containing the squares of numbers from 1 to 20 (inclusive).

The function uses a **generator** expression to create the tuple, which is more memory efficient than creating a list and then converting it to a tuple. The function doesn't take any parameters. When the function is called and the tuple is returned, it is then printed to the console.

## Exercise 38

With a given tuple (1,2,3,4,5,6,7,8,9,10), write a program to print the first half values in one line and the last half values in one line.

### Code

```
tup = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
lt = int(len(tup) / 2)
print(tup[:lt], tup[lt:])
```

### ChatGPT Explanation

The code creates a tuple `tup` containing the numbers 1 to 10. Then it creates a variable `lt` that is the integer division of the length of the tuple by 2.

Finally, it prints out the first half of the tuple by slicing it from the start to the index of `lt` and the second half of the tuple by slicing it from the index of `lt` to the end.

The final output would be (1, 2, 3, 4, 5) (6, 7, 8, 9, 10).

## Exercise 39

Write a program to generate and print another tuple whose values are even numbers in the given tuple (1,2,3,4,5,6,7,8,9,10).

### Code

```
tpl = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
tpl1 = tuple(i for i in tpl if i % 2 == 0)
print(tpl1)
```

### ChatGPT Explanation

This code creates a tuple called `tpl` that contains the integers 1 through 10. It then creates a new tuple called `tpl1` that contains only the even numbers from `tpl` using a list comprehension.

It iterates over each element in `tpl` and checks if it is divisible by 2 using the modulo operator. If it is, it adds it to the new tuple `tpl1`.

Finally, it prints out the new tuple `tpl1`, which should contain the numbers 2, 4, 6, 8, and 10.

## Exercise 40

Write a program which accepts a string as input to print "Yes" if the string is "yes" or "YES" or "Yes", otherwise print "No".

### Code

```
input = input("Enter string:")
output = ""
output = output.join(
    ["Yes" if input == "yes" or input == "YES" or input == "Yes" else "No"]
)
print(str(output))
```

### ChatGPT Explanation

This code snippet prompts the user to enter a string and then assigns the input to a variable called `input`.

It then uses a `join` method to join a list of strings together. The list is created using a list comprehension that checks if the input is equal to "yes", "YES", or "Yes", if it is it returns "Yes" otherwise it returns "No".

The final result is then stored in the `output` variable and printed as a string. The output will be "Yes" if the user entered "yes", "YES", or "Yes" and "No" for any other input.

## Exercise 41

Write a program which can `map()` to make a list whose elements are square of elements in `[1,2,3,4,5,6,7,8,9,10]`.

### Code

```
li = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
squaredNumbers = map(lambda x: x ** 2, li) # returns map type object data
print(list(squaredNumbers)) # converting the object into list
```

### ChatGPT Explanation

The code above uses the `map()` function to apply the lambda function `lambda x: x ** 2` to each element in the list `li`.

The lambda function takes one argument `x` and returns the square of `x`. The `map()` function returns a map object, which is an iterator over the results of applying the lambda function to each element in the list. The `list()` function is then used to convert the map object into a list, so that the squared numbers can be printed using the `print()` function.

The output will be the list of squared numbers of the elements in the original list.

## Exercise 42

Write a program which can `map()` and `filter()` to make a list whose elements are square of even number in `[1,2,3,4,5,6,7,8,9,10]`.

### Code

```
def even(x):
    return x % 2 == 0

def squer(x):
    return x * x

li = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
li = map(
    squer, filter(even, li)
) # first filters number by even number and the apply map() on the resultant elements
print(list(li))
```

### ChatGPT Explanation

The code defines two functions: `even(x)` and `squer(x)`. The `even` function takes a single argument `x` and returns `True` if `x` is even and `False` otherwise. The `square` function takes a single argument `x` and returns `x` squared.

Then it creates a list of numbers called `li`. It then uses the `map()` function to apply the `square` function to each element in the list. But before applying the `square` function, it applies the `filter` function to the list, which filters out all the elements that are not even.

So the final output will be the square of even numbers of the list.

## Exercise 43

Write a program which can `filter()` to make a list whose elements are even number between 1 and 20 (both included).

### Code

```
def even(x):  
    return x % 2 == 0  
  
evenNumbers = filter(even, range(1, 21))  
print(list(evenNumbers))
```

### ChatGPT Explanation

The code defines a function called `even(x)` that takes in a parameter `x` and returns `True` if `x` is even, and `False` if `x` is odd.

Then, it uses the `filter()` function to filter out all the even numbers in the range of 1 to 21 and assigns the resulting filter object to the variable `evenNumbers`.

Finally, it converts the filter object into a list and prints it. So the output will be a list of all even numbers between 1 and 21.

## Exercise 44

Write a program which can `map()` to make a list whose elements are square of numbers between 1 and 20 (both included).

### Code

```
def sqr(x):  
    return x * x  
  
squaredNumbers = list(map(sqr, range(1, 21)))  
print(squaredNumbers)
```

### ChatGPT Explanation

In this code snippet, we are defining a function `sqr` that takes in a single argument `x` and returns the square of `x` by multiplying `x` with itself.

Next, we use the `map` function to apply the `sqr` function to every element in the range 1 to 21 (exclusive). This creates a map object, which is a lazy evaluation of the function applied to each element in the range.

To obtain the list of the squared numbers, we use the `list` function to convert the map object into a list and store it in the variable `squaredNumbers`.

Finally, we print the `squaredNumbers` list to see the output, which is the list of squares of numbers from 1 to 20.



## Exercise 45

Define a class named `American` which has a static method called `printNationality`.

### Code

```
class American:
    @staticmethod
    def printNationality():
        print("I am American")

american = American()
american.printNationality()
American.printNationality()
```

### ChatGPT Explanation

The code defines a class `American` in Python. The class has a static method called `printNationality()`. This method simply prints the string "I am American".

A object `american` is created from the `American` class. The object `american` invokes the `printNationality()` method using the object instance `american.printNationality()`.

The class `American` directly invokes the static method `printNationality()` by calling `American.printNationality()`.

Both of these method invocations produce the same output "I am American".

## Exercise 46

Define a class named `American` and its subclass `NewYorker`.

### Code

```
class American:
    pass

class NewYorker(American):
    pass

american = American()
newyorker = NewYorker()

print(american)
print(newyorker)
```

### ChatGPT Explanation

The code defines two classes `American` and `NewYorker` in Python. The `American` class is an empty class which doesn't have any attributes or methods. The `NewYorker` class is a subclass of `American` class.

In the code, two objects `american` and `newyorker` are created. The first object `american` is an instance of the `American` class, while the second object `newyorker` is an instance of the `NewYorker` class.

When the `print` statement is executed, the object `american` returns the memory location of the `american` object. The object `newyorker` returns the memory location of the `newyorker` object.

The output will be:

```
<__main__.American object at 0x...>
<__main__.NewYorker object at 0x...>
```

## Exercise 47

Define a class named `Circle` which can be constructed by a radius. The `Circle` class has a method which can compute the area.

### Code

```
class Circle:
    def __init__(self, r):
        self.radius = r

    def area(self):
        return 3.1416 * (self.radius ** 2)

circle = Circle(5)
print(circle.area())
```

### ChatGPT Explanation

The code defines a class `Circle` with a constructor `__init__` method that takes a radius `r` as input and sets it as an instance variable `radius` of the object. The class also has a method `area` which calculates and returns the area of the circle using the formula  $3.1416 * (\text{self.radius} ** 2)$ .

Finally, an object `circle` of class `Circle` is created with radius 5 and its area is calculated by calling `circle.area()`. The result of the area calculation is then printed.

## Exercise 48

Define a class named `Rectangle` which can be constructed by a length and width. The `Rectangle` class has a method which can compute the area.

### Code

```
class Rectangle:
    def __init__(self, l, w):
        self.length = l
        self.width = w

    def area(self):
        return self.length * self.width

rect = Rectangle(2, 4)
print(rect.area())
```

### ChatGPT Explanation

This code defines a class `Rectangle` in Python. The class has a constructor method `__init__` which initializes two instance variables `length` and `width` with the values passed as parameters when a `Rectangle` object is created. The class also has a method `area` which calculates and returns the area of the rectangle by multiplying the `length` and `width` of the object.

Finally, the code creates an instance of the `Rectangle` class and assigns it to the variable `rect`. The area of the rectangle is then printed by calling the `area` method on the `rect` object. The output of the code will be `8`.

## Exercise 49

Define a class named `Shape` and its subclass `Square`. The `Square` class has an `init` function which takes a `length` as argument. Both classes have a `area` function which can print the area of the shape where `Shape`'s area is 0 by default.

### Code

```
class Shape:
    def __init__(self):
        pass

    def area(self):
        return 0

class Square(Shape):
    def __init__(self, length=0):
        Shape.__init__(self)
        self.length = length

    def area(self):
        return self.length * self.length

Asqr = Square(5)
print(Asqr.area())

print(Square().area())
```

### ChatGPT Explanation

The code defines a class `Shape` that has an `__init__` method which takes no arguments, and an `area` method that returns 0.

It also defines a class `Square` that inherits from the `Shape` class. The `Square` class has an `__init__` method that takes an optional argument `length` and initializes the `length` attribute of the object. The `Square` class also overrides the `area` method of the `Shape` class to return the area of a square, which is calculated as the square of its length.

The code then creates an object `Asqr` of the `Square` class with the length 5 and prints its area by calling the `area` method on the object. The code also creates another object of the `Square` class with no arguments and prints its area, which is 0 because the length was not specified.

## Exercise 50

Please raise a `RuntimeError` exception.

### Code

```
raise RuntimeError("something wrong")
```

### ChatGPT Explanation

The code raises a `RuntimeError` exception with the message "something wrong". When this exception is raised, the program execution stops and an error message is displayed.

## Exercise 51

Write a function to compute  $5/0$  and use `try/except` to catch the exceptions.

### Code

```
def divide():
    return 5 / 0

try:
    divide()
except ZeroDivisionError as ze:
    print("Why on earth you are dividing a number by ZERO!!")
except:
    print("Any other exception")
```

### ChatGPT Explanation

The code block implements a simple `try-except` block to handle exceptions raised during the execution of the `divide` function.

The `divide` function returns the result of dividing 5 by 0, which will raise a `ZeroDivisionError` exception.

In the `try` block, we call the `divide` function, and in the following `except` block, we handle the `ZeroDivisionError` exception by printing a custom error message: "Why on earth you are dividing a number by ZERO!!".

In case any other exception is raised, we catch it in another `except` block and print a different message: "Any other exception".

This way, if an exception is raised during the execution of the `divide` function, it will be caught and handled, and the program will not crash.

## Exercise 52

Define a custom exception class which takes a string message as attribute.

### Code

```
class CustomException(Exception):
    """Exception raised for custom purpose

    Attributes:
        message -- explanation of the error
    """

    def __init__(self, message):
        self.message = message

num = int(input())
try:
    if num < 10:
        raise CustomException("Input is less than 10")
    elif num > 10:
        raise CustomException("Input is grater than 10")
except CustomException as ce:
    print("The error raised: " + ce.message)
```

### ChatGPT Explanation

The code defines a custom exception class called `CustomException` that inherits from the built-in `Exception` class. The custom exception class takes in a message as an argument and sets it as an attribute.

The code then takes in a number from the user, converts it to an integer, and stores it in the `num` variable. The code then uses a `try-except` block to handle the custom exception.

If `num` is less than 10, the code raises the custom exception with the message "Input is less than 10". If `num` is greater than 10, the code raises the custom exception with the message "Input is greater than 10".

In the except block, the code prints the message attribute of the custom exception, with the message "The error raised: " concatenated to the front.



## Exercise 53

Assuming that we have some email addresses in the "username@companyname.com" format, please write program to print the user name of a given email address. Both user names and company names are composed of letters only.

### Code

```
import re

email = "john@google.com elise@python.com"
pattern = "(\\w+)@\\w+\\.com"
ans = re.findall(pattern, email)
print(ans)
```

### ChatGPT Explanation

The given code uses the `re` module in Python to find all occurrences of a specific pattern in a string. The pattern being searched for is defined as `"(\\w+)@\\w+\\.com"` and it represents an email address format.

The `findall` method of the `re` module is used to search the string `email` and return all occurrences of the pattern defined in `pattern`. The result is stored in the `ans` variable and finally, the result is printed using the `print` statement.

The output of the code will be `['john', 'elise']`, which are the substrings that match the pattern.

## Exercise 54

Assuming that we have some email addresses in the "username@companyname.com" format, please write program to print the company name of a given email address. Both user names and company names are composed of letters only.

### Code

```
import re

email = "john@google.com elise@python.com"
pattern = "\w+@(\w+).com"
ans = re.findall(pattern, email)
print(ans)
```

### ChatGPT Explanation

The code imports the `re` (regular expression) module. Then it defines a string `email` that contains two email addresses.

A regular expression pattern `pattern` is then defined to match words (`\w+`) before the `@` symbol, followed by another word (`\w+`) and `.com`.

The `re.findall` function is called, passing in the `pattern` and `email` strings. The function returns a list of all the matches found in the `email` string and it is assigned to the variable `ans`.

Finally, `ans` is printed, which would output: `['google', 'python']`.

## Exercise 55

Write a program which accepts a sequence of words separated by whitespace as input to print the words composed of digits only.

### Code

```
email = input().split()
ans = [word for word in email if word.isdigit()] # using list comprehension method
print(ans)
```

### ChatGPT Explanation

This code takes a list of words as input and splits it into separate words. Then it uses a list comprehension to extract all the words that are numbers. Finally, it prints the resulting list of numbers.

## Exercise 56

Print a unicode string "hello world".

### Code

```
s = u"hello world"  
print(s)
```

### ChatGPT Explanation

The following code is written in Python, it defines a Unicode string `s` with the value "hello world" and then prints it to the console.

Unicode strings in Python are denoted by prefixing the string with the letter 'u' as in `u"hello world"`. This means that the string is encoded in the Unicode character set, which allows for the representation of a wide range of characters from many different languages.

In this code snippet, `s` is defined as a Unicode string and then printed to the console using the `print` function. The output of the code will be `hello world`.

## Exercise 57

Write a program to read an ASCII string and to convert it to a unicode string encoded by utf-8.

### Code

```
s = input()
u = s.encode("utf-8")
print(u)
```

### ChatGPT Explanation

The code first takes input from the user and stores it in the variable `s`. Then it uses the `encode` method of the `str` class to encode the string `s` into `utf-8` format and stores the result in a variable `u`. Finally, it prints the encoded string `u`.

The `encode` method is used to convert the string into a specific encoding format, in this case, `utf-8`. The `utf-8` format is a widely used character encoding that can handle a wide range of characters from different languages and scripts.

By converting the string into `utf-8`, it can be transmitted over the internet and stored in various data storage systems in a consistent manner, ensuring that the characters are properly displayed on different platforms.

## Exercise 58

Write a special comment to indicate a Python source code file is in unicode.

### Code

```
# -*- coding: utf-8 -*-
```

### ChatGPT Explanation

The line `# -*- coding: utf-8 -*-` is a special comment line in the Python code that indicates the encoding of the source code file. In this case, the source code is encoded using UTF-8, which is a widely used character encoding that can handle a large number of characters from various scripts and languages.

This line is usually placed at the beginning of the file to specify the encoding of the file, which helps the Python interpreter understand and process the file correctly.

## Exercise 59

Write a program to compute  $1/2+2/3+3/4+\dots+n/n+1$  with a given  $n$  input by console ( $n \geq 0$ ).

### Code

```
n = int(input())
sum = 0
for i in range(1, n + 1):
    sum += i / (i + 1)
print(round(sum, 2)) # rounded to 2 decimal point
```

### ChatGPT Explanation

The code snippet computes and prints the sum of a series. The series formula is defined as follows:

$$\text{sum} = (1/2) + (2/3) + (3/4) + \dots + (n/(n + 1))$$

The input  $n$  is expected to be a positive integer and determines the number of terms in the series. The code uses a `for` loop to iterate through the range 1 to  $n + 1$  and calculates each term of the series by dividing  $i$  by  $(i + 1)$  where  $i$  is the current iteration value. Finally, the sum of all terms is calculated and rounded to 2 decimal places using the `round` function. The result is then printed.

## Exercise 60

Write a program to compute:  $f(n) = f(n - 1) + 100$ ,  $n > 0$  with a given  $n$  input by console ( $n; 0$ ).

### Code

```
def f(n):
    if n == 0:
        return 0
    return f(n - 1) + 100

n = int(input())
print(f(n))
```

### ChatGPT Explanation

The following code is a recursive function in Python that calculates the sum of the series  $100 + 200 + 300 + \dots + 100 * n$  where  $n$  is the input argument.

The function  $f(n)$  takes an integer  $n$  as input. If  $n$  is 0, it returns 0. Otherwise, it calls itself with the argument  $n-1$  and adds 100 to the result. This process repeats until the base case of  $n=0$  is reached, and then the final result is returned up the call stack.

The code also takes a user input  $n$ , converts it to an integer, and passes it as an argument to the  $f$  function. The result of the  $f$  function is then printed.



## Exercise 61

The Fibonacci Sequence is computed based on the following formula:

```
f(0)=0,  
f(1)=1,  
f(n) = f(n - 1) + f(n - 2), n > 1
```

Please write a program to compute the value of  $f(n)$  with a given  $n$  input by console.

### Code

```
def f(n):  
    if n < 2:  
        return n  
    return f(n - 1) + f(n - 2)  
  
n = int(input())  
print(f(n))
```

### ChatGPT Explanation

This code implements the recursive function  $f(n)$ , which returns the  $n$ -th number in the Fibonacci sequence. The Fibonacci sequence is a sequence of numbers where each number is the sum of the two preceding numbers. In this case, the sequence starts with 0 and 1, so the first two numbers in the sequence are 0 and 1.

The function first checks if  $n$  is less than 2, and if so, it returns  $n$ . This serves as the base case for the recursion. If  $n$  is greater than or equal to 2, the function calls itself twice, each time with an argument that is one less than the current value of  $n$ . The two calls to  $f(n - 1)$  and  $f(n - 2)$  represent the two preceding numbers in the sequence, and the sum of these two calls gives the next number in the sequence. This process continues until  $n$  becomes less than 2, at which point the base case is reached and the recursion stops.

Finally, the code takes an input  $n$  from the user, and passes it as an argument to the function  $f(n)$ . The function returns the  $n$ -th number in the Fibonacci sequence, which is printed to the screen.

## Exercise 62

The Fibonacci Sequence is computed based on the following formula:

```
f(0)=0,  
f(1)=1,  
f(n) = f(n - 1) + f(n - 2), n > 1
```

Please write a program to compute the value of f(n) with a given n input by console and prints all the values all along.

### Code

```
def f(n):  
    if n < 2:  
        fibo[n] = n  
        return fibo[n]  
    fibo[n] = f(n - 1) + f(n - 2)  
    return fibo[n]  
  
n = int(input())  
fibo = [0] * (n + 1) # initialize a list of size (n+1)  
f(n) # call once and it will set value to fibo[0-n]  
fibo = [str(i) for i in fibo] # converting integer data to string type  
ans = ",".join(fibo) # joining all string element of fibo with ',' character  
print(ans)
```

### ChatGPT Explanation

The code above calculates the first n terms of the Fibonacci sequence using a recursive function f.

The function takes an argument n and checks if it is less than 2. If it is, it stores n in the fibo list and returns n.

Otherwise, it calculates the n-th term of the sequence by calling the function with n-1 and n-2 as arguments, adds the results and stores it in the fibo list. The function returns the n-th term.

The function is called with the input value of n and the fibo list is initialized with 0's. The function is called to calculate the first n terms of the sequence and the results are stored in the fibo list.

Finally, the list is converted to a string with each element separated by a comma and the result is printed.

## Exercise 63

Please write a program using generator to print the even numbers between 0 and n in comma separated form while n is input by console.

### Code

```
def EvenGenerator(n):
    i = 0
    while i <= n:
        if i % 2 == 0:
            yield i
        i += 1

n = int(raw_input())
values = []
for i in EvenGenerator(n):
    values.append(str(i))

print ",".join(values)
```

### ChatGPT Explanation

The following code generates a list of even numbers up to a given number `n`. This is achieved using a generator function `EvenGenerator(n)`. The generator function `EvenGenerator(n)` generates even numbers up to `n` using a `while` loop.

The `yield` statement is used in the generator function to return each even number. This allows the generator to pause its execution after each iteration and return the current value of `i`. This way, we can generate a series of even numbers without having to store them all in memory.

The generator function is called with the input `n` and the generated even numbers are stored in the list `values`. The `values` list is then converted to a string separated by commas using the `join` method and printed as the final result.

## Exercise 64

Please write a program using generator to print the numbers which can be divisible by 5 and 7 between 0 and n in comma separated form while n is input by console.

### Code

```
def generate(n):
    for i in range(n + 1):
        if (
            i % 35 == 0
        ): # 5*7 = 35, if a number is divisible by a & b then it is also divisible by a*b
            yield i

n = int(input())
resp = [str(i) for i in generate(n)]
print(",".join(resp))
```

### ChatGPT Explanation

The following code snippet defines a function called `generate` that takes an integer `n` as input. The function uses the `range` function to generate a range of numbers from 0 to `n` and uses a for loop to iterate through this range.

The `for` loop checks if the number `i` is divisible by 35, if it is divisible, it is added to the generator using the `yield` keyword. This allows us to generate a sequence of numbers that are divisible by 35.

Finally, the list comprehension `resp` takes the numbers generated from the generator and converts them into a list of strings. The list is then joined using the `join` method and separated by a comma. The final result is then printed to the console.

## Exercise 65

Please write assert statements to verify that every number in the list [2,4,6,8] is even.

### Code

```
data = [2, 4, 5, 6]
for i in data:
    assert i % 2 == 0, "{} is not an even number".format(i)
```

### ChatGPT Explanation

This code is checking if each element in the list `data` is even or not. If the element is not even, it will raise an `AssertionError` with the message "`x is not an even number`", where `x` is the current element in the iteration.

The `assert` statement is used to check if a given condition is met, and if not, raise an exception. In this case, the condition being checked is if the element divided by 2 has a remainder of 0 (i.e. if it is even). If this condition is met for all elements in the list, the code will execute without raising any exceptions.

## Exercise 66

Please write a program which accepts basic mathematic expression from console and print the evaluation result.

### Code

```
expression = input()
ans = eval(expression)
print(ans)
```

### ChatGPT Explanation

The code reads an expression as input from the user and then evaluates it using the `eval` function in Python. The result of the evaluation is then printed to the console.

For example, if the user inputs `2 + 3`, the code will evaluate `2 + 3` to `5` and then print `5` to the console.

## Exercise 67

Please write a binary search function which searches an item in a sorted list. The function should return the index of element to be searched in the list.

### Code

```
def binary_search_Ascending(array, target):
    lower = 0
    upper = len(array)
    print("Array Length:", upper)
    while lower < upper:
        x = (lower + upper) // 2
        print("Middle Value:", x)
        value = array[x]
        if target == value:
            return x
        elif target > value:
            lower = x
        elif target < value:
            upper = x
```

```
Array = [1, 5, 8, 10, 12, 13, 55, 66, 73, 78, 82, 85, 88, 99]
print("The Value Found at Index:", binary_search_Ascending(Array, 82))
```

### ChatGPT Explanation

The code performs a binary search algorithm in an ascending sorted array to find the index of a target value.

The function `binary_search_Ascending` takes two arguments, `array` and `target`. It first initializes two variables, `lower` and `upper`, to be the start and end of the array, respectively. The `while` loop runs until `lower` becomes greater than or equal to `upper`. In each iteration, `x` is calculated as the middle value of the current subarray defined by `lower` and `upper`, and the value of the array at that index is stored in `value`.

Then, the `target` value is compared with the `middle` value, if they are equal, the index `x` is returned as the answer. If the `target` value is greater than the `middle` value, `lower` is updated to be equal to `x` so that the next search will be in the upper half of the array. If the `target` value is less than the `middle` value, `upper` is updated to be equal to `x`, so that the next search will be in the lower half of the array.

The `print` statements are used for debugging purposes to see the intermediate values of `x` and the length of the array.

The code then calls the function with an example array and target value and prints the index at which the target value is found.



## Exercise 68

Please generate a random float where the value is between 10 and 100 using Python module.

### Code

```
import random

rand_num = random.uniform(10, 100)
print(rand_num)
```

### ChatGPT Explanation

The code is generating a random floating-point number between 10 and 100 using the `uniform` function from the `random` module in Python. The `uniform` function generates a random float value between two given parameters (inclusive). In this case, it's generating a random float value between 10 and 100.

## Exercise 69

Please generate a random float where the value is between 5 and 95 using Python module.

### Code

```
import random

rand_num = random.uniform(5, 95)
print(rand_num)
```

### ChatGPT Explanation

The code generates a random floating-point number between 5 and 95.

The first line `import random` is used to import the `random` module in Python, which provides functions for generating random numbers.

The second line `rand_num = random.uniform(5, 95)` generates a random floating-point number between 5 and 95 (including 5 but not including 95) and assigns it to the variable `rand_num`.

Finally, the line `print(rand_num)` prints the generated random number.

## Exercise 70

Please write a program to output a random even number between 0 and 10 inclusive using random module and list comprehension.

### Code

```
import random

resp = [i for i in range(0, 11, 2)]
print(random.choice(resp))
```

### ChatGPT Explanation

The code imports the `random` module in Python, which provides a suite of functions to generate pseudo-random numbers.

Then, a list `resp` is created using list comprehension, with the numbers from 0 to 10 with a step of 2.

Finally, the `random.choice` method is used to select a random element from the list `resp`. The selected random element is then printed to the console. This code will output a random even number between 0 and 10.

## Exercise 71

Please write a program to output a random number, which is divisible by 5 and 7, between 10 and 150 inclusive using random module and list comprehension.

### Code

```
import random

resp = [i for i in range(10, 151) if i % 35 == 0]
print(random.choice(resp))
```

### ChatGPT Explanation

The code imports the `random` module in Python, which provides functions for generating pseudo-random numbers.

Then, it defines a list called `resp` using a list comprehension that generates a list of numbers from 10 to 150 (inclusive) that are divisible by 35 (i.e., have a remainder of 0 when divided by 35).

Finally, the code prints a random element from this list using the `random.choice` function, which returns a randomly selected element from a given sequence (in this case, the `resp` list).

## Exercise 72

Please write a program to generate a list with 5 random numbers between 100 and 200 inclusive.

### Code

```
import random

resp = random.sample(range(100, 201), 5)
print(resp)
```

### ChatGPT Explanation

This code snippet is written in Python and uses the `random` module to generate a list of 5 random numbers within the range of 100 to 200 (inclusive).

First, we import the `random` module, which provides a suite of functions for generating pseudo-random numbers.

Next, we use the `random.sample` method to generate a list of 5 random numbers within the range of 100 to 200. The `random.sample` method takes two arguments: the first argument is the range of numbers to choose from (in this case, it's `range(100, 201)`), and the second argument is the number of items to choose from that range (in this case, it's 5).

Finally, we use the `print` function to output the list of 5 randomly generated numbers.

## Exercise 73

Please write a program to randomly generate a list with 5 even numbers between 100 and 200 inclusive.

### Code

```
import random

resp = random.sample(range(100, 201, 2), 5)
print(resp)
```

### ChatGPT Explanation

This code snippet generates 5 random numbers from a range of even numbers between 100 and 201 (inclusive).

It starts by importing the `random` module, which contains a number of useful functions for generating random numbers, shuffling lists, and choosing random items from a list.

Next, the line `resp = random.sample(range(100, 201, 2), 5)` generates a list of 5 random numbers from a range of even numbers between 100 and 201 (inclusive).

The `range(100, 201, 2)` function generates a list of all even numbers between 100 and 201 (inclusive), and `random.sample(range(100, 201, 2), 5)` randomly chooses 5 numbers from this list.

Finally, the line `print(resp)` prints the list of 5 random numbers.

## Exercise 74

Please write a program to randomly generate a list with 5 numbers, which are divisible by 5 and 7 , between 1 and 1000 inclusive.

### Code

```
import random

lst = [i for i in range(1, 1001) if i % 35 == 0]
resp = random.sample(lst, 5)
print(resp)
```

### ChatGPT Explanation

This code snippet imports the `random` module in Python, which provides functions to generate random numbers.

The first line defines a list called `lst` which contains all the numbers from 1 to 1000 that are divisible by 35. This is done using a list comprehension.

The second line uses the `random.sample` function from the `random` module to generate 5 random numbers from the list `lst`. This function takes two arguments, the first is the list from which the random samples should be selected, and the second is the number of samples to select.

Finally, the last line prints the result of the `resp` variable, which contains the 5 random numbers selected from the `lst` list.

## Exercise 75

Please write a program to randomly print a integer number between 7 and 15 inclusive.

### Code

```
import random
print(random.randrange(7, 16))
```

### ChatGPT Explanation

The following code snippet is written in Python and it uses the `random` module.

The `random.randrange(start, stop)` method is used to generate a random number within a range, in this case between 7 and 16 (excluding 16).

The result of the `random.randrange(7, 16)` method is then printed to the console. This means that each time the code is run, a different random number between 7 and 16 will be generated and printed to the console.



## Exercise 76

Please write a program to compress and decompress the string "hello world!hello world!hello world!hello world!".

### Code

```
import zlib

t = zlib.compress(b'hello world!hello world!hello world!hello world!')
print(t)
print(zlib.decompress(t))
```

### ChatGPT Explanation

The code is using the `zlib` library to perform data compression and decompression. The `zlib` library provides in-memory compression and decompression functions, including the `compress` and `decompress` functions.

In the code, the string "hello world!hello world!hello world!hello world!" is first converted to a byte object using `b'hello world!hello world!hello world!hello world!'`. This byte object is then passed to the `compress` function of the `zlib` library which returns a compressed version of the byte object.

The compressed version of the byte object is then printed, and finally, the compressed version of the byte object is passed to the `decompress` function of the `zlib` library which returns the original byte object. The decompressed byte object is then printed.

In this way, the code demonstrates the use of `zlib` library for data compression and decompression in Python.

## Exercise 77

Please write a program to print the running time of execution of "1+1" for 100 times.

### Code

```
import time

before = time.time()
for i in range(100):
    x = 1 + 1
after = time.time()
execution_time = after - before
print(execution_time)
```

### ChatGPT Explanation

The code is measuring the execution time of a simple operation. It's using the `time` module in Python which provides various time-related functions.

At the start of the code, the current time is recorded by `before = time.time()`.

Then, the code runs a for loop 100 times where a simple operation `x = 1 + 1` is performed.

At the end of the loop, the current time is recorded again by `after = time.time()`.

Finally, the execution time is calculated by subtracting `before` from `after` and stored in the variable `execution_time`. The execution time is then printed using `print(execution_time)`. This shows how much time it took to run the simple operation 100 times.

## Exercise 78

Please write a program to shuffle and print the list [3,6,7,8].

### Code

```
import random

lst = [3, 6, 7, 8]
random.shuffle(lst)
print(lst)
```

### ChatGPT Explanation

This code is in the Python programming language. It makes use of the `random` module which provides a set of functions for generating pseudo-random numbers.

The code defines a list called `lst` that contains the values [3, 6, 7, 8].

Then, the function `shuffle` from the `random` module is used to shuffle the items in the list `lst`. The `shuffle` function randomly reorders the items in the list.

Finally, the shuffled list is printed using the `print` statement.

## Exercise 79

Please write a program to generate all sentences where subject is in ["I", "You"] and verb is in ["Play", "Love"] and the object is in ["Hockey", "Football"].

### Code

```
subjects = ["I", "You"]
verbs = ["Play", "Love"]
objects = ["Hockey", "Football"]

for sub in subjects:
    for verb in verbs:
        for obj in objects:
            print("{} {} {}".format(sub, verb, obj))
```

### ChatGPT Explanation

The following code snippet in Python generates all possible combinations of a sentence using elements from three different lists: `subjects`, `verbs`, and `objects`.

The `subjects` list contains the words "I" and "You". The `verbs` list contains the words "Play" and "Love". The `objects` list contains the words "Hockey" and "Football".

The code uses a nested loop, with each outer loop iterating through each element in one of the lists (`subjects`, `verbs`, `objects`). The inner loop then iterates through the elements of the other two lists, combining each subject-verb-object combination and printing the resulting sentence.

For example, the first combination would be "I Play Hockey". The code will generate a total of  $(2 * 2 * 2 =) 8$  combinations of sentences in total.

## Exercise 80

Please write a program to print the list after removing even numbers in [5,6,77,45,22,12,24].

### Code

```
li = [5, 6, 77, 45, 22, 12, 24]
lst = list(filter(lambda n: n % 2 != 0, li))
print(lst)
```

### ChatGPT Explanation

This is a Python code snippet that uses the `filter` method and `lambda` function to extract the odd numbers from a list.

The list `li` is initialized with a set of integers. The `filter` method and `lambda` function are used to extract only the odd numbers from the list.

The `filter` method takes a function and a list as arguments. The `lambda` function is an anonymous function that takes an argument `n` and returns `True` if `n` is not divisible by 2, and `False` otherwise. The `filter` method then applies the `lambda` function to each element of the `li` list and returns a new list containing only the elements for which the function returns `True`. This new list is stored in the `lst` variable.

Finally, the resulting list is printed using the `print` function.

## Exercise 81

By using list comprehension, please write a program to print the list after removing numbers which are divisible by 5 and 7 in [12,24,35,70,88,120,155].

### Code

```
li = [12, 24, 35, 70, 88, 120, 155]
li = [x for x in li if x % 35 != 0]
print(li)
```

### ChatGPT Explanation

The code defines a list of integers called `li`. Then, it uses a list comprehension to create a new list of integers, filtering out the elements of `li` that are divisible by 35.

The `if x % 35 != 0` part in the list comprehension checks whether `x` is not divisible by 35. If this condition is true, then `x` is added to the new list.

Finally, the code prints the new filtered list.

## Exercise 82

By using list comprehension, please write a program to print the list after removing the 0th, 2nd, 4th,6th numbers in [12,24,35,70,88,120,155].

### Code

```
li = [12, 24, 35, 70, 88, 120, 155]
li = [li[i] for i in range(len(li)) if i % 2 != 0]
print(li)
```

### ChatGPT Explanation

This code snippet uses Python to create a list `li` of numbers, and then creates a new list `li` with only the elements located at odd indices of the original list `li`.

It starts by defining a list `li` of 7 integers.

Then, it creates a new list `li` using a list comprehension. The list comprehension includes all elements located at odd indices (indexes 1, 3, 5, etc.) of the original list `li`.

Finally, the new list `li` is printed to the console.

## Exercise 83

By using list comprehension, please write a program to print the list after removing the 2nd - 4th numbers in [12,24,35,70,88,120,155].

### Code

```
li = [12, 24, 35, 70, 88, 120, 155]
li = [li[i] for i in range(len(li)) if i < 3 or 4 < i]
print(li)
```

### ChatGPT Explanation

This code snippet is written in Python and it defines a list called `li` which contains 7 numbers.

The next line of code creates a new list called `li` which contains elements of the original list `li` based on the conditions specified in a list comprehension.

A list comprehension is a concise way of creating a new list from an existing one.

The condition specified in this list comprehension is `if i < 3 or 4 < i`, which means the elements will be selected if the index of the element (`i`) is less than 3 or greater than 4.

Finally, the new list is printed using the `print()` function.



## Exercise 84

By using list comprehension, please write a program generate a 3x5x8 3D array whose each element is 0

### Code

```
array = [[[0 for col in range(8)] for col in range(5)] for row in range(3)]
```

### ChatGPT Explanation

The code creates a 3-dimensional list in Python, also called an array. This array has 3 layers. The first layer has 3 rows. The second layer has 5 columns. The third layer has 8 columns. Each value in the 3rd layer is set to zero.

In other words, the code creates a 3x5x8 matrix filled with zeros. The matrix has 3 pages, with each page having 5 rows and 8 columns, making a total of  $3 \times 5 \times 8 = 120$  zeros in the array.

## Exercise 85

By using list comprehension, please write a program to print the list after removing the 0th, 4th, 5th numbers in [12,24,35,70,88,120,155].

### Code

```
li = [12, 24, 35, 70, 88, 120, 155]
li = [li[i] for i in range(len(li)) if i not in (0, 4, 5)]
print(li)
```

### ChatGPT Explanation

This code defines a list of numbers named `li`. It then creates a new list named `li` which consists of elements from the original `li` list where the index is not equal to 0, 4, and 5.

The new `li` is constructed using a list comprehension. A list comprehension is a concise way to create a new list in Python. The syntax is:

```
new_list = [expression for item in iterable if condition]
```

In this code, expression is `li[i]`, iterable is `range(len(li))`, and condition is `i not in (0, 4, 5)`. This expression generates a new list with elements from the `li` list where the index is not 0, 4, or 5. Finally, the code prints the new `li`.

## Exercise 86

By using list comprehension, please write a program to print the list after removing the value 24 in [12,24,35,24,88,120,155].

### Code

```
li = [12, 24, 35, 24, 88, 120, 155]
li.remove(24) # this will remove only the first occurrence of 24
print(li)
```

### ChatGPT Explanation

This code defines a list named `li` with some values. Then, it removes the first occurrence of the value 24 from the list using the `remove()` method. Finally, it prints the resulting list.

## Exercise 87

With two given lists [1,3,6,78,35,55] and [12,24,35,24,88,120,155], write a program to make a list whose elements are intersection of the above given lists.

### Code

```
list1 = [1, 3, 6, 78, 35, 55]
list2 = [12, 24, 35, 24, 88, 120, 155]
set1 = set(list1)
set2 = set(list2)
intersection = set.intersection(set1, set2)
print(intersection)
```

### ChatGPT Explanation

The code defines two lists, `list1` and `list2` with some values. Then, it creates two sets `set1` and `set2` from these two lists respectively.

The `intersection` variable is assigned the result of the intersection of `set1` and `set2` using the `intersection` method of the `set` class. This method returns a set that contains all the elements that are present in both `set1` and `set2`.

Finally, the code prints the `intersection` variable which contains the result of the intersection of `set1` and `set2`.

## Exercise 88

With a given list [12,24,35,24,88,120,155,88,120,155], write a program to print this list after removing all duplicate values with original order reserved.

### Code

```
li = [12, 24, 35, 24, 88, 120, 155, 88, 120, 155]
for i in li:
    if li.count(i) > 1:
        li.remove(i)
print(li)
```

### ChatGPT Explanation

This code snippet creates a list of integers called `li` and then loops through each element `i` in the list.

For each iteration of the loop, it checks if the count of `i` in the list is greater than 1. If the count is greater than 1, it removes the first occurrence of `i` from the list using the `remove` method.

Finally, it prints the updated list after all the duplicates have been removed.

## Exercise 89

Define a class `Person` and its two child classes: `Male` and `Female`. All classes have a method `getGender` which can print `"Male"` for `Male` class and `"Female"` for `Female` class.

### Code

```
class Person(object):
    def getGender(self):
        return "Unknown"

class Male(Person):
    def getGender(self):
        return "Male"

class Female(Person):
    def getGender(self):
        return "Female"

aMale = Male()
aFemale = Female()
print(aMale.getGender())
print(aFemale.getGender())
```

### ChatGPT Explanation

The code creates a Python class called `Person` with a method `getGender` that returns the string `"Unknown"`.

The class `Male` is a subclass of the `Person` class, and it overrides the `getGender` method to return the string `"Male"`. Similarly, the class `Female` is a subclass of the `Person` class and it overrides the `getGender` method to return the string `"Female"`.

Next, the code creates two objects, `aMale` and `aFemale`, from the classes `Male` and `Female`, respectively. Finally, the code prints the result of calling the `getGender` method on each object. This will print `"Male"` and `"Female"`.

## Exercise 90

Please write a program which count and print the numbers of each character in a string input by console.

### Code

```
import string

s = input()
for letter in string.ascii_lowercase:
    cnt = s.count(letter)
    if cnt > 0:
        print("{} , {}".format(letter, cnt))
```

### ChatGPT Explanation

This code is a program that takes in a string as input from the user and then counts the number of occurrences of each letter in the string (case-insensitive).

It uses the `string` module's `ascii_lowercase` constant to get a list of all lowercase letters in the alphabet. The program loops over each letter in `ascii_lowercase`, and for each letter, it counts the number of occurrences of that letter in the input string `s` using the `count` method. If the count of a letter is greater than 0, the program prints the letter and its count in the format "`{}` , `{}`".

## Exercise 91

Please write a program which accepts a string from console and print it in reverse order.

### Code

```
s = input()
s = "".join(reversed(s))
print(s)
```

### ChatGPT Explanation

This code takes an input string from the user and reverses its characters. The `input()` function is used to get the string from the user. Then, the `reversed` function is used to reverse the order of characters in the string.

Finally, the `join` function is used to join the reversed characters together into a new string, which is then printed to the console.



## Exercise 92

Please write a program which accepts a string from console and print the characters that have even indexes.

### Code

```
s = "H1e2l3l4o5w6o7r8l9d"
s = [s[i] for i in range(len(s)) if i % 2 == 0]
print("".join(s))
```

### ChatGPT Explanation

The given code takes a string `s` and creates a new list of characters by picking characters at even-indexed positions in the input string `s`.

The final output is a concatenation of all the characters in the new list, represented as a string. The result is printed on the console.

## Exercise 93

Please write a program which prints all permutations of [1,2,3]

### Code

```
import itertools
print(list(itertools.permutations([1, 2, 3])))
```

### ChatGPT Explanation

This code uses the `itertools` module in Python to generate all possible permutations of a given list `[1, 2, 3]`.

The `permutations` function from the `itertools` module takes a list as input and returns an iterable object containing all possible arrangements (permutations) of the elements in that list.

In this code, the permutations are computed for the list `[1, 2, 3]`, and the result is then converted to a list using the `list` function.

The final output of the code will be a list of all possible permutations of the list `[1, 2, 3]`.

## Exercise 94

Write a program to solve a classic ancient Chinese puzzle: We count 35 heads and 94 legs among the chickens and rabbits in a farm. How many rabbits and how many chickens do we have?

### Code

```
def solve(numheads, numlegs):
    ns = "No solutions!"
    for i in range(numheads + 1):
        j = numheads - i
        if 2 * i + 4 * j == numlegs:
            return i, j
    return ns, ns

numheads = 35
numlegs = 94
solutions = solve(numheads, numlegs)
print(solutions)
```

### ChatGPT Explanation

The code defines a function named `solve` that takes two parameters `numheads` and `numlegs`. It then uses a `for` loop to iterate over a range of values starting from 0 to the value of `numheads + 1`. Within the loop, it calculates the value of `j` by subtracting the current value of `i` from `numheads`. Then, it checks if the result of the expression `2 * i + 4 * j` is equal to `numlegs`. If it is, it returns a tuple of values `i` and `j`.

If the loop completes and no solutions are found, the function returns a tuple `ns, ns` which contains the string "No solutions!".

Finally, the code sets `numheads` to 35 and `numlegs` to 94, then calls the `solve` function passing these values. The returned solution is then printed.

## Exercise 95

Given the participants' score sheet for your University Sports Day, you are required to find the runner-up score. You are given scores. Store them in a list and find the score of the runner-up.

If the following string is given as input to the program: - 5 2 3 6 6 5

Then, the output of the program should be: - 5

### Code

```
arr = map(int, input().split())
arr = list(set(arr))
arr.sort()
print(arr[-2])
```

### ChatGPT Explanation

The code first takes an input from the user of a list of integers separated by spaces. The input is then passed to the `map` function which converts the strings into integers and returns a `map` object.

Next, the `map` object is converted into a list using the `list` function, and duplicates are removed by converting the list into a set and then back into a list.

Finally, the list is sorted in ascending order and the second last element is printed which is the second largest element in the list.

## Exercise 96

You are given a string *S* and width *W*. Your task is to wrap the string into a paragraph of width.

If the following string is given as input to the program: - ABCDEFGHIJKLM-NOQRSTUVWXYZ 4

Then, the output of the program should be: - ABCD EFGH IJKL IMNO QRST  
UVWX YZ

### Code

```
import textwrap

string = input()
width = int(input())

print(textwrap.fill(string, width))
```

### ChatGPT Explanation

The code snippet is using the `textwrap` module to wrap a given text string into multiple lines.

The user inputs the text string and the width (number of characters per line) they want to wrap the text into.

The code then calls the `fill` method from the `textwrap` module and passes in the text string and the desired width as arguments. This method returns the text string, wrapped into multiple lines so that no line exceeds the width specified.

Finally, the wrapped text is printed to the console.

## Exercise 97

You are given a date. Your task is to find what the day is on that date.

### Code

```
import calendar

month, day, year = map(int, input().split())
dayId = calendar.weekday(year, month, day)
print(calendar.day_name[dayId].upper())
```

### ChatGPT Explanation

This code is using the `calendar` module in Python to find the day of the week for a given date.

First, it takes three inputs from the user: `month`, `day`, and `year`. These three inputs are combined into a single string, then separated and converted into integers using the `map` function and the `int` function.

Next, the `calendar.weekday` function is used to determine the weekday for the given date. This function takes three arguments: the year, the month, and the day. The result is an integer representing the day of the week, where Monday is 0 and Sunday is 6.

Finally, the `calendar.day_name` array is used to convert the integer representation of the day of the week into a string, and the `upper` function is used to convert the string to uppercase. The resulting string is then printed.

## Exercise 98

Given 2 sets of integers, M and N, print their symmetric difference in ascending order. The term symmetric difference indicates those values that exist in either M or N but do not exist in both.

### Code

```
n = int(input())
set1 = set(map(int, input().split()))

m = int(input())
set2 = set(map(int, input().split()))

ans = list(set1 ^ set2)
ans.sort()
for i in ans:
    print(i)
```

### ChatGPT Explanation

This code takes two sets of integers as input and finds the symmetric difference between them. The symmetric difference of two sets is defined as the set of elements that are in either of the sets, but not in their intersection.

1. The first input line `n` is the number of elements in set 1, and the next line is the elements of `set1`, separated by spaces. These elements are converted to integers using the `map` function and stored in `set1`.
2. The next two lines are similar, with `m` being the number of elements in `set2`, and the next line being the elements of `set 2`. These elements are stored in `set2`.
3. The symmetric difference between the two sets is found using the `^` operator, and the result is stored in the list `ans`.
4. The list `ans` is sorted and the elements are printed one by one.

## Exercise 99

You are given words. Some words may repeat. For each word, output its number of occurrences. The output order should correspond with the input order of appearance of the word. See the sample input/output for clarification.

If the following string is given as input to the program: - 4 bcdef abcdefg bcde bcdef

Then, the output of the program should be: - 3 2 1 1

### Code

```
n = int(input())

word_list = []
word_dict = {}

for i in range(n):
    word = input()
    if word not in word_dict:
        word_list.append(word)
        word_dict[word] = word_dict.get(word, 0) + 1

print(len(word_list))
for word in word_list:
    print(word_dict[word], end=" ")
```

### ChatGPT Explanation

This code takes in an integer `n` as input which represents the number of words to be entered. Then it takes `n` words as input and stores them in a list `word_list`. It also creates a dictionary `word_dict` where the key is a word and the value is the frequency of that word in the input words.

It then checks if a word is not already present in the dictionary, if yes, it appends the word to the list `word_list`. It then increments the value of the word in the dictionary by 1.

In the end, it prints the length of the list `word_list` and the frequency of each word in `word_dict` separated by a space.



## Exercise 100

You are given a string. Your task is to count the frequency of letters of the string and print the letters in descending order of frequency.

If the following string is given as input to the program: - aabbccde

Then, the output of the program should be: - b 3 a 2 c 2 d 1 e 1

### Code

```
s = list(input())

dict_count_ = {k: s.count(k) for k in s}
list_of_tuples = [(k, v) for k, v in dict_count_.items()]
list_of_tuples.sort(key=lambda x: x[1], reverse=True)

for item in list_of_tuples:
    print(item[0], item[1])
```

### ChatGPT Explanation

The following code in Python takes an input string, `s`, and performs the following actions:

1. Creates a dictionary, `dict_count_`, where each character in the string is a key and its value is the count of that character in the string.
2. Creates a list of tuples, `list_of_tuples`, where each tuple consists of a character and its count.
3. Sorts the list of tuples in descending order of the count of characters.
4. Prints the character and its count for each tuple in the sorted list.

## How was this document created?

Programming is learned by programming. For this reason we need a set of explained exercises and program them in order to learn how to program. This document was created with this premise in mind.

To generate the content of this document we used two resources:

1. A pre-compiled list of Python exercises and
2. The ChatGPT Large Language Model

The exercises in this documents were taken from the Break The Ice With Python GitHub repository. The repository contains more than 100 simple Python exercises aimed for beginners.

ChatGPT is a revolutionary large language model. It is capable of generating text on literally any subject. The model is quite good in generating explanations of code snippets. This serves as a good tool to teach programming as demonstrated in this document.

To obtain the code explanations we prompted ChatGPT with the following prompt:

“ Explain me the following code snippet written in Python as explaining it to someone who doesn't know programming in Python “

after which the coding exercise in Python followed.

All the exercises can be found in this GitHub repository. Read more about this project on [iSquared.digital](https://iSquared.digital)