Sum of possitive numbers

Calculate the sum of all possitive numbers in a list.
Example: for the list [2, -1, 3], the expected sum is 2 + 3 = 5

In [2]:

```python
a = [2, -1, 3]
# YOUR CODE HERE
# Hint:
# - Loop through elements of the list
# - Use conditional statement to check if an element is possitive
# - If yes, add to the final sum

sum = 0
for i in a:
        if (i>0) == True:
            sum += i

print(f"sum = {sum}")
```

```
sum = 5
```

Now, use the code that you've developed to create a function that takes in a list and returns the sum of positive numbers.

In [3]:

```python
def sum_positives(a_list):
    # identify the positive numbers and sum them up
    sum = 0
    for i in a_list:
        if (i>0) == True:
            sum += i
    return sum

# example
a = [2, -1, 3]
print(f"The sum of positive numbers is {sum_positives(a)}")
```

```
The sum of positive numbers is 5
```

Call the function with the list `a` above.

Body Mass Index

Let's evaluate the Body Mass Index (BMI) of a user based on a given weight and height

Body Mass Index is a simple calculation using a person’s height and weight. The formula is BMI = kg/m2 where kg is a person’s weight in kilograms and m2 is their height in metres squared.

A BMI of 25.0 or more is overweight, while the healthy range is 18.5 to 24.9. BMI applies to most adults 18-65 years.

instruction:

- write this program in a .py file.
- open the file BMI.py
- ask users to input their height, weight, and age
- make sure that they input their height in meter and if they entered in cm, then convert it to meter.
- check if the user is eligible for this index based on their age.
- calculate the BMI (only if eligible)
- based on BMI send approaprate message to users whether they are over-weighted, under-weighted or normal.
- ask the users if they want to exit (Y|N)?
- countinue asking for height, weight and age if the answer is No (N).

In [7]:

```
# YOUR CODE
# This program calculates body mass index of users using their weight and height.

import os

os.system("clear")

def calculate_bmi():
    while True:
        try:
            # input height, weight and age
            height = float(input("Enter your height in meters (or in cm): "))
            weight = float(input("Enter your weight in kilograms: "))
            age = int(input("Enter your age: "))

            # convert height to meter level if input is in centimeter level
            if height > 3:
                height = height / 100

            # calculate bmi
            if 18 <= age <= 65:
                bmi = weight / (height ** 2)

                if bmi < 18.5:
                    print(f"Your BMI is {bmi:.2f}. You are underweight.")
                elif 18.5 <= bmi <= 24.9:
                    print(f"Your BMI is {bmi:.2f}. You are within the normal weight range.")
                else:
                    print(f"Your BMI is {bmi:.2f}. You are overweight.")
            else:
                print("BMI calculation is only applicable for people aged 18-65.")

            # define exit program
            exit_program = input("Do you want to exit? (Y/N): ")
            if exit_program == 'Y':
                print("Exiting the program. Thank you!")
                break
            elif exit_program != 'N':
                print("Invalid input. Please enter 'Y' for yes or 'N' for no.")
                break

        except ValueError:
            print("Invalid input. Please enter valid numbers for height, weight, and age.")


if __name__ == "__main__":
    calculate_bmi()
```

```
Enter your height in meters (or in cm): 185
Enter your weight in kilograms: 74
Enter your age: 22
Your BMI is 21.62. You are within the normal weight range.
Do you want to exit? (Y/N): N
Enter your height in meters (or in cm): 175
Enter your weight in kilograms: 89
Enter your age: 23
Your BMI is 29.06. You are overweight.
Do you want to exit? (Y/N): Y
Exiting the program. Thank you!
```

Fibonacci

The Fibonacci series starts with 0 and 1. The next number is the sum of the last two numbers.
$x_0 = 0, x_1 = 1, x_{n+1} = x_n + x_{n-1}$

- Write a function `get_Fibonacci_number` to compute x_n of the Fibonacci series.

E.g: - `get_Fibonacci_number(0)` returns 0 - `get_Fibonacci_number(1)` returns 1 - `get_Fibonacci_number(3)` returns 2

- Approximate the golden ratio:

$$\lim_{x \to \infty} \frac{fib_x}{fib_(x-1)}$$

Hint: you can use loops, generators functions, or recursive functions. For the golden ratio, you can think of a big enouph number (e.g. 1000) to represent infinity.

- create a list 100 golden ratio for the first 100 fibonacci numbers and plot the result using the matplotlib library.

In [11]:

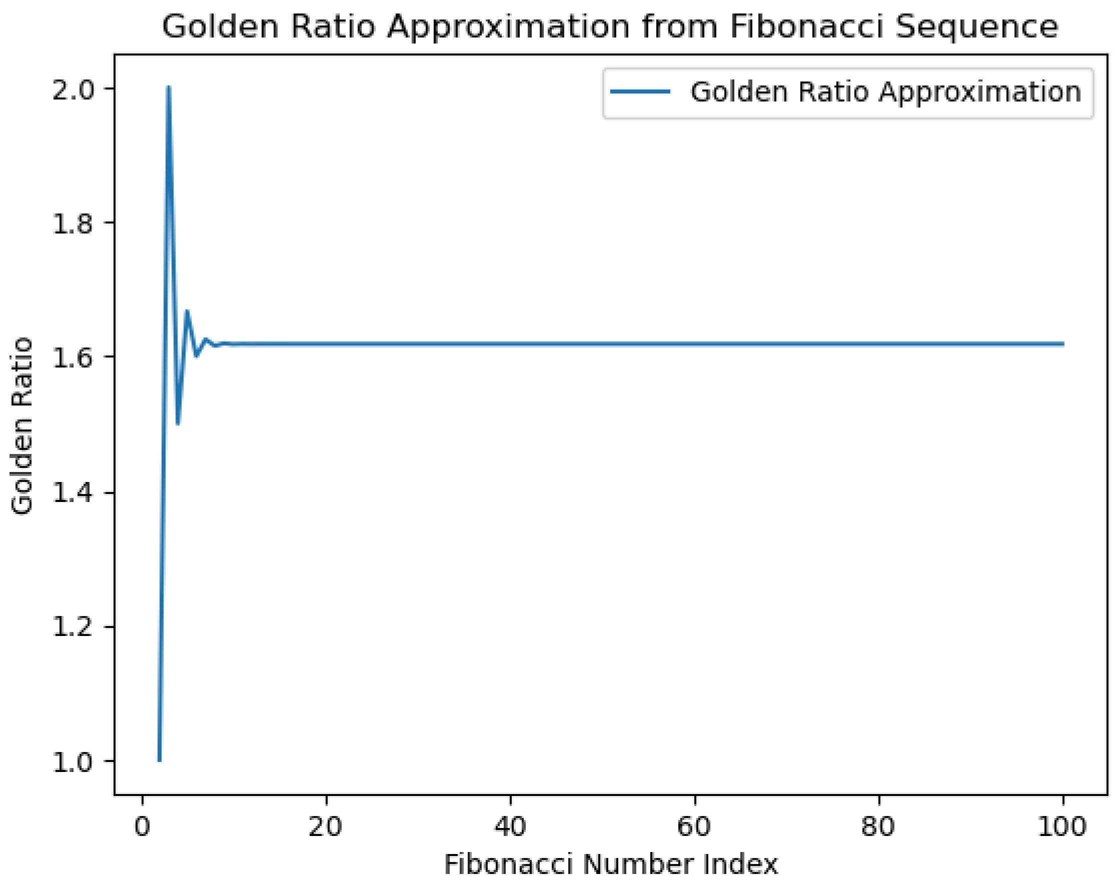
```python
import matplotlib.pyplot as plt

def get_Fibonacci_number(n):
    # get the fibonacci number
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        fib_0, fib_1 = 0, 1
        for i in range(2, n+1):
            fib_0, fib_1 = fib_1, fib_0 + fib_1
        return fib_1

def get_golden_ratios(n):
    # get the golden ratio
    golden_ratios = []
    for i in range(2, n+1):
        fib_n = get_Fibonacci_number(i)
        fib_n_minus_1 = get_Fibonacci_number(i-1)
        golden_ratio = fib_n / fib_n_minus_1
        golden_ratios.append(golden_ratio)
    return golden_ratios

n = 100
golden_ratios = get_golden_ratios(n)

# plot the result
plt.plot(range(2, n+1), golden_ratios, label="Golden Ratio Approximation")
plt.title('Golden Ratio Approximation from Fibonacci Sequence')
plt.xlabel('Fibonacci Number Index')
plt.ylabel('Golden Ratio')
plt.legend()
plt.show()
```



Write a function to get the largest Fibonacci number that is equal or smaller than a given number.
For example:

- Given 2, the functions should return 2
- Given 10, the functions should return 8

In [4]:

```
def largest_fibonacci_leq(n):
    small_number = 0
    large_number = 1

    # find the lower and upper fibonacci boundaries of the input number
    while large_number <= n:
        small_number = large_number
        large_number = small_number + large_number

    return small_number

input_number = int(input("Please input a number:"))
print(largest_fibonacci_leq(input_number))
```

```
Please input a number:52
32
```

Dictionary

A Python ditionary comprises of student numbers as keys and student names as values. Write a function to capitalize all the student names in the dictionary.

In [3]:

```
def capitalize_student_names(student_dict):
    # capitalize the names in dict
    capitalized_dict = {number: name.capitalize() for number, name in student_dict.items()}
    return capitalized_dict

# example for function
students = {1: 'luiz',
            2: 'li',
            3: 'patrick'
           }

capitalized_students = capitalize_student_names(students)
print(capitalized_students)
```

```
{1: 'Luiz', 2: 'Li', 3: 'Patrick'}
```

Character counts

Write a function that count the frequencies of each alphabet character in a given string. The function should return a dictionary, in which each key is a character and each value is the corresponding frequency. All characters are treated as their lowercases,

meaning 'E' is the same as 'e'.
For example: Calling the function for 'Hello' will return {'h': 1, 'e': 1, 'l': 2, 'o': 1}.

In [22]:

```
def character_frequencies(s):
    # convert string to all lowercases
    s = s.lower()

    frequency_dict = {}

    # count frequency of characters
    for char in s:
        if char.isalpha():
            if char in frequency_dict:
                frequency_dict[char] += 1
            else:
                frequency_dict[char] = 1

    return frequency_dict

result = character_frequencies('Hello')
print(result)
```

```
{'h': 1, 'e': 1, 'l': 2, 'o': 1}
```

Extrema (Optional)

Given a list of numbers representing a series, count how many time the values change their trends, i.e. from increasing to descreasing and vi versa.
Examples of these changes are:

- [0, 2, 1]
- [0, -2, -2, 3]

In [9]:

```
def count_trend_changes(series):
    # if there is not enough elements, return 0
    if len(series) < 2:
        return 0

    trend_changes = 0
    current_trend = None

    # identify the trend
    for i in range(len(series) - 1):
        if series[i] < series[i + 1]:
            trend = 'increasing'
        elif series[i] > series[i + 1]:
            trend = 'decreasing'
        else:
            continue

        # count changes if the trend changes
        if current_trend and trend != current_trend:
            trend_changes += 1

        current_trend = trend

    return trend_changes

print(count_trend_changes([0, 2, 1]))
print(count_trend_changes([0, -2, -2, 3]))
```

```
1
1
```

Approximate π (Optional)

In [25]:

```
from random import random
```

One method to approximate the value of π is through simulation. Given the function `random` generates a number in the range $[0, 1]$ randomly, write a function to approxmiate π.

Hints:

- π is the area of a cirle with radius of 1.

- For any random point in the unit square (positions top-right of the origin), the change of this point belonging to the quarter unit circle is $\pi/4$

In [8]:

```python
import random
def approximate_pi(num_samples):
    inside_circle = 0

    # generate random points in the circle
    for _ in range(num_samples):
        x = random.random()
        y = random.random()

        # count if the point is in the circle
        if x**2 + y**2 <= 1:
            inside_circle += 1

    # 4*pi = inside_circle/num_samples
    pi_estimate = (inside_circle / num_samples) * 4
    return pi_estimate

num_samples = 10000000
pi_approx = approximate_pi(num_samples)
print(f"Estimated value of π: {pi_approx}")
```

```
Estimated value of π: 3.1415504
```

Prime Number (Optional)

Recieve a number N from users and return if the number is a Prime number. A prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. In other words, they have only 2 factors: 1 and themselves. Display relavent message if the user entered anything except integer.

Hint: If the number N is devisible by any natural numbers between 2 and the first natural number bigger than $\sqrt{N}$, the the number is **NOT** a Prime number.

In [4]:

```
import math
from math import sqrt

def is_prime_number(n):
    # Identify whether n is a prime number

    if n <= 1:
        return False
    elif n == 4:
        return False
    else:
        for i in range(2, math.ceil(sqrt(n)), 1):
            if n % i == 0:
                return False

    return True

def check_prime():
    # Check whether the imput number is a prime number

    user_input = input("Enter a number: ")

    try:
        number = int(user_input)
    except ValueError:
        print("The input is not a valid integer.")
        return

    if is_prime_number(number):
        print(f"{number} is a prime number.")
    else:
        print(f"{number} is not a prime number.")

check_prime()
```

```
Enter a number: 53
53 is a prime number.
```