Problem 2. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

$$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...$$

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

Programming Knowledge required: How to write loops, How to compute Fibonacci's numbers in linear time.

Solution Outline: Fibonacci numbers are a sequence of numbers that follow the following recurrence:

$$f(n) = f(n-1) + f(n-2)$$

Where $f(n)$ represents the n^{th} Fibonacci number. To write a program that computes n^{th} Fibonacci number can be easily done by a recursive implementation. But the recursive implementation is too slow, since it recomputes many values again and again. Hence we will implement an iterative version of it which computes any Fibonacci term only once, hence saving computation time.

We will initialize two variables namely `a`, `b` which are initialized to 1, 2 respectively. We will also initialize another variable which keep tracks of sum of even Fibonacci terms, let it be `even_sum` and is initialized to 0. Then we will run an infinite loop until `a` exceeds four million. While looping we will check if `a` is an even number. If yes then we will increment `even_sum` by that value. Finally `even_sum` contains the final answer.

Python Solution

```python
a, b = 1, 2
even_sum = 0

while a <= int(4e6):
    if a % 2 == 0:
        even_sum += a
    a, b = b, a + b

print(even_sum)
```