**3. Iteration (Unconditional loops)**

**Task 2:**

**Plan:**

1. Create list variable to store Fibonacci sequence, and previous values.
2. For loop; Loop through I in range (1,11).
3. Calculate next value in Fibonacci sequence.
4. Append calculated value to Fibonacci list.
5. If there has been 10 iterations end for loop. Else repeat from step 3.
6. Output content of Fibonacci list.

**Pseudocode:**

Fibonacci 🡸 [0,1]

PreviousValues 🡸 [0,1]

For I in range (1,11):

PreviousValues 🡸 previousValues[1], sum(previousValues)

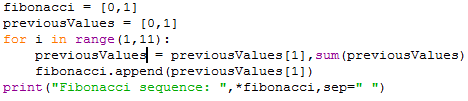
Fibonacci.append(previousValues[1])

Output ("Fibonacci sequence: ",\*fibonacci)

**Variables table:**

|  |  |  |
| --- | --- | --- |
| Variable name | Data type | Comment |
| Fibonacci | List | Stores fibonacci sequence. |
| previousValues | List | Calculates and temporarily stores the next value in the fibonacci sequence. |

**Screenshot evidence:**

In the screenshot to the left, a list variable called “fibonacci” is defined with the first 2 values of the fibonacci sequence (0,1). A second list called “previousValues” is also defined with the same values of (0,1). In the screenshot below is an example ****of 2 different ways you could define the variables. In my solution I separated the 2 variables on different lines, as it makes it easier to read. After you have defined the variables, a for loop then passes through 10 iterations. Calculating and appending the next number in the sequence, by adding the last 2. Finally, the first 10 numbers in the Fibonacci sequence are output.

**Sample run (Output):**



The screenshot above shows my solution running in the python idle, the first 10 numbers of the Fibonacci sequence have been output.

**Flow charts:**

START

previousValues 🡸 previousValues[1], sum(previousValues)

previousValues 🡸 [0,1]

fibonacci.append(previousValues[1])

Output ("Fibonacci sequence: ", \*fibonacci)

Yes

No

For I in range (1,11). Has I passed 10 iterations

fibonacci 🡸 [0,1]