Introduction to Programming

PYTHON

Chapter 5 – Command 01: for … loop

Presenter: **Dr. Nguyen Dinh Long**

Email: dinhlonghcmut@gmail.com

Google-site: https://sites.google.com/view/long-dinh-nguyen

Oct. 2022

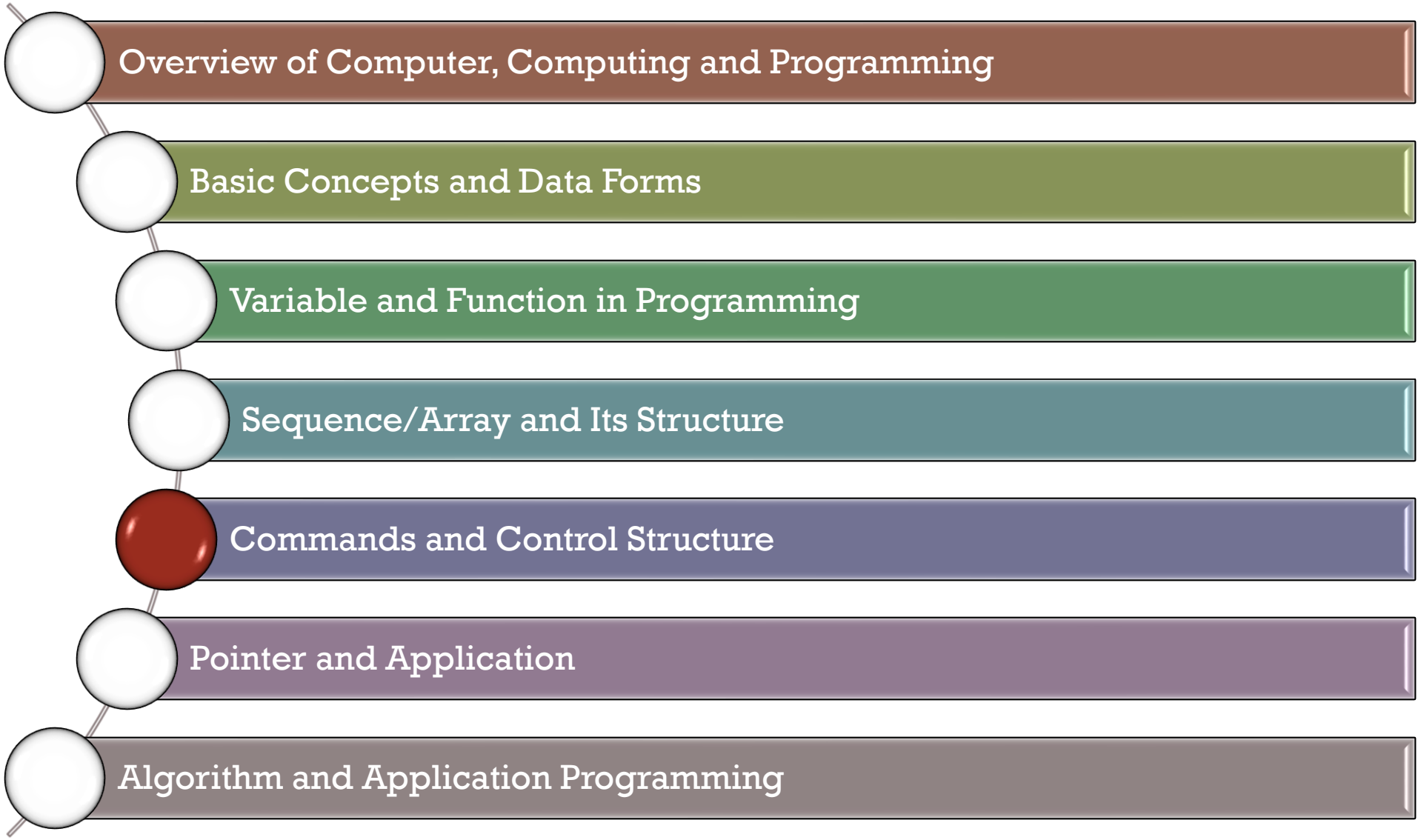
Programming

Hamilton was a self-taught programmer, working in the US in the 1960's. Owing to the success of her previous work, Hamilton was the first programmer to be hired for the Apollo project. She became the Director of Software Engineering at the MIT Instrumentation lab. Her lab developed the on-board flight software for NASA's Apollo space project, which took humankind to the moon.

The achievement was a monumental task at a time when computer technology was in its infancy: The astronauts had access to only 72 kilobytes of computer memory (a 256-gigabyte cell phone today carries almost a million times more storage space). Programmers had to use paper punch cards to feed information into room-sized computers with no screen interface.

Margaret Hamilton, NASA's lead software engineer for the Apollo, stands next to the code she wrote by hand that took humanity to the moon in 1969.

Outline


Overview of Computer, Computing and Programming
Basic Concepts and Data Forms
Variable and Function in Programming
Sequence/Array and Its Structure
Commands and Control Structure
Pointer and Application
Algorithm and Application Programming

References

Main:

- Maurizio Gabbrielli and Simone Martini, 2010. *Programming Languages: Principles and Paradigms*, Springer.
- Cao Hoàng Trụ, 2004. *Ngôn ngữ lập trình- Các nguyên lý và mô hình*, Nhà xuất bản Đại học Quốc gia Tp. Hồ Chí Minh

More:

- Wes McKinney, 2013. *Python for Data Analysis*, O'Reilly Media.
- Guido van Rossum, Fred L. Drake, Jr.,, 2012. *The Python Library Reference*, Release 3.2.3.

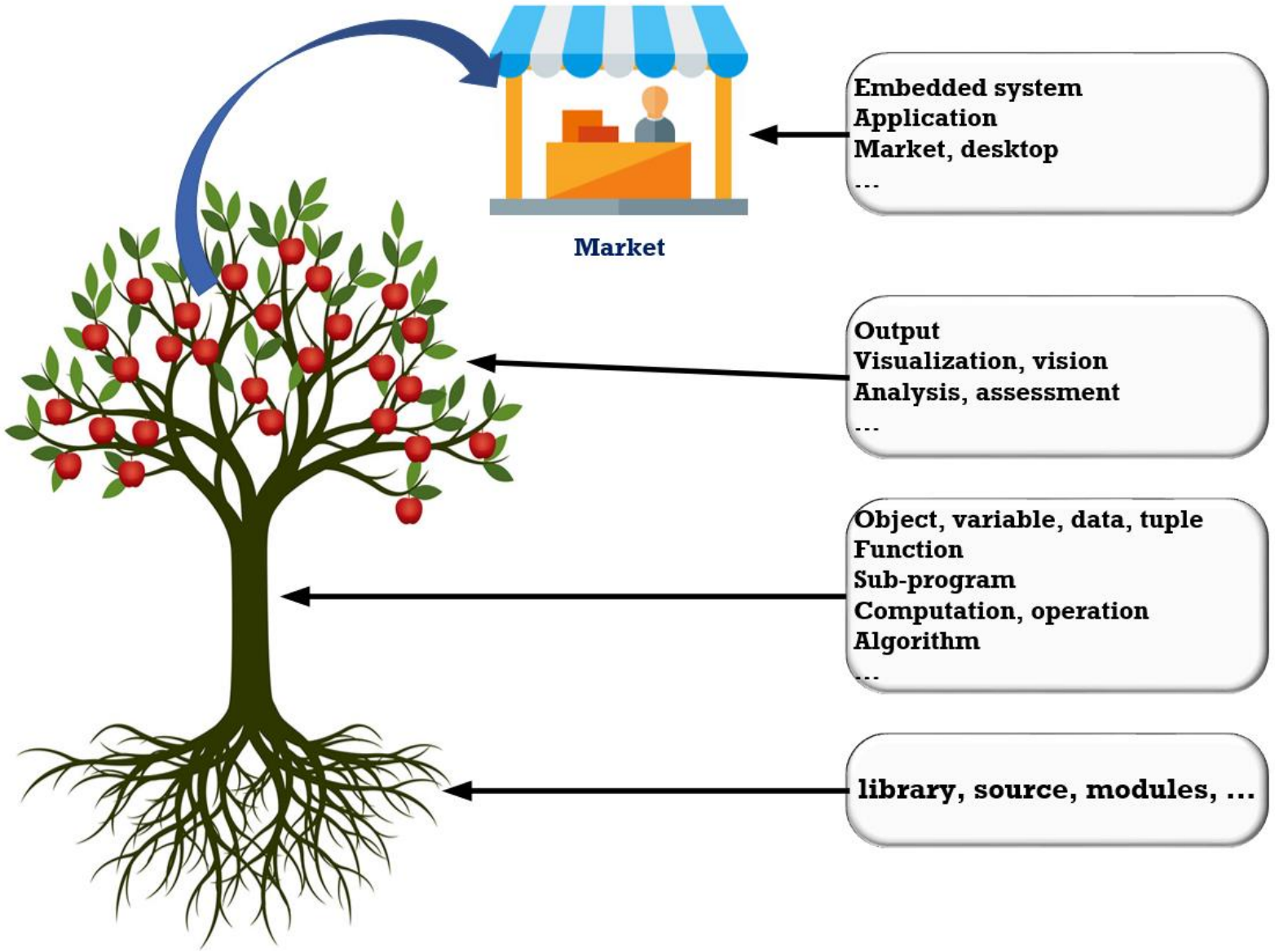
- Slides here are collected and modified from several sources in Universities and Internet.

Content of Chapter 5

1. Commands in Python programming
2. Control programs: “for ...” loop and Examples (1 week)
3. Control programs: “while ... do" loop and Examples (1 week)
4. Control programs: “if ... else" loop and Examples (1 week)
5. Examples and Practices: Combine Python loops (1 week)

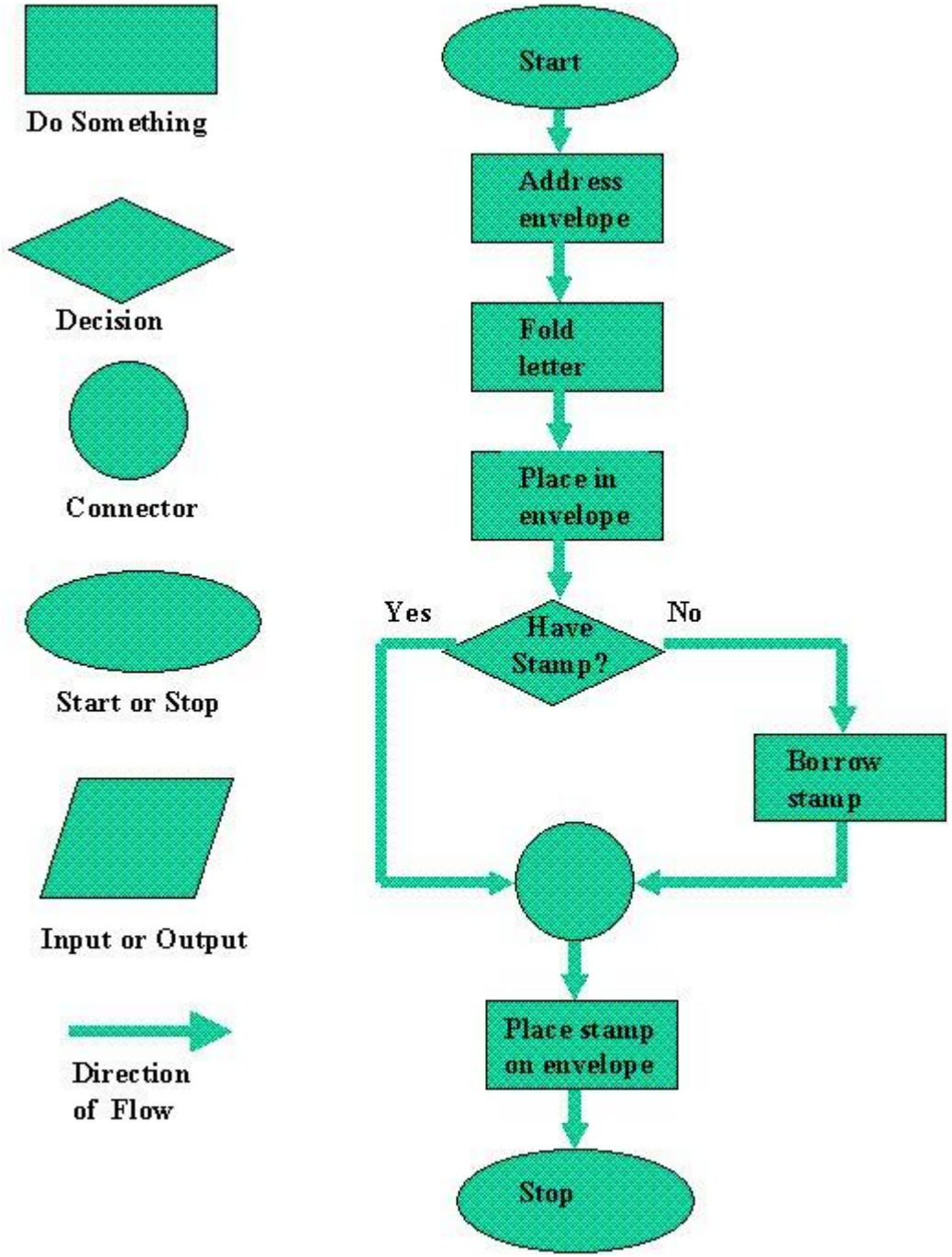
Computer programs

- **General structure:**



Structure of Computer programs

- **Computer programming:**
  - Objects
  - Types
  - Variables
  - Methods
  - Tuples



Part I

For ... Loop in python programming

For ... Loop in Python

❑ **Python For loop:**

- The for loop in Python is an iterating function. If you have a sequence object like a String, Tuple, List, Set or Dictionary and Array, you can use the for loop to iterate over the items contained within the list.

- The functionality of the for loop is not very different from what you see in multiple other programming languages.

- We explore the Python for loop in detail and learn to iterate over different sequences including lists, tuples, array and more. Additionally, we'll learn to control the flow of the loop using the break and continue statements.

For ... Loop in Python

❑ **Basic Syntax of the Python for loop:**

The basic syntax of the for loop in Python looks something similar to the one mentioned below.

```
for itarator_variable in sequence_name:
        Statements
        . . .
        Statements
```

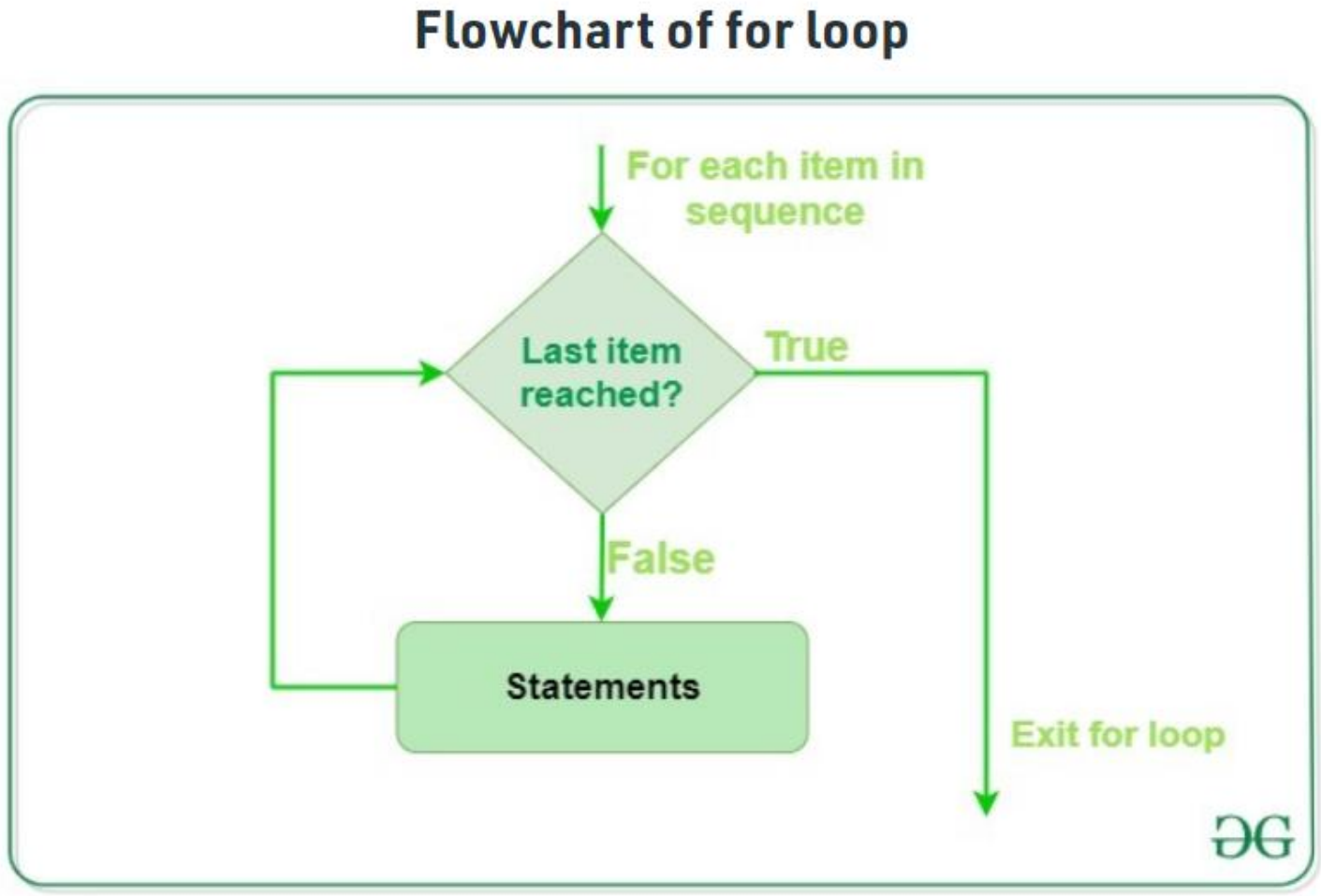
- The first word of the statement starts with the **keyword "for"** which signifies the beginning of the for loop.
- Then we have the **iterator variable** which iterates over the sequence and can be used within the loop to perform various functions
- The next is the **"in" keyword** in Python which tells the iterator variable to loop for elements within the sequence
- We have the **sequence variable** which can either be a list, a tuple, or any other kind of iterator.
- The statements part of the loop is where you can play around with the iterator variable and perform various function

For ... Loop in Python

- **Basic Syntax of the Python for loop:**

```
for itarator_variable in sequence_name:
        Statements
        . . .
        Statements
```

```
for var in iterable:
    # statements
```



For ... Loop in Python

- **Using the for loop to iterate over a Python list or tuple:**

Lists and Tuples are iterable objects. Let's look at how we can loop over the elements within these objects now.

```
words= ["Apple", "Banana", "Car", "Dolphin" ]
for word in words:
        print (word)
```

Output:

```
Apple
Banana
Car
Dolphin
```

```
nums = (1, 2, 3, 4)

sum_nums = 0

for num in nums:
    sum_nums = sum_nums + num

print(f'Sum of numbers is {sum_nums}')

# Output
# Sum of numbers is 10
```

For ... Loop in Python

- **Print individual letters of a string using the for loop:**

Python string is a sequence of characters. If within any of your programming applications, you need to go over the characters of a string individually, you can use the for loop here.

```
word="anaconda"
for letter in word:
        print (letter)
```

Output:

```
a
n
a
c
o
n
d
a
```

The reason why this loop works is because Python considers a “string” as a sequence of characters instead of looking at the string as a whole.

For ... Loop in Python

- **Using For Loops in Python Dictionary:**

Python3

```
# Iterating over dictionary
print("Dictionary Iteration")
d = dict()
d['xyz'] = 123
d['abc'] = 345
for i in d:
    print("% s % d" % (i, d[i]))
```

Output:

```
Dictionary Iteration
xyz  123
abc  345
```

For ... Loop in Python

❑ **Using For Loops in Python Set:**

We will take a set of item, and iterate over the each of the items using for loop.

```
myset = {'python', 'programming', 'examples'}

for x in myset:
    print(x)
```

Output

```
python
examples
programming
```

For ... Loop in Python

❑ Nesting Python for loops:

- When we have a for loop inside another for loop, it's called a nested for loop. There are multiple applications of a nested for loop.
- Consider the list example above. The for loop prints out individual words from the list. But what if we want to print out the individual characters of each of the words within the list instead?
- This is where a nested for loop works better. The first loop (parent loop) will go over the words one by one. The second loop (child loop) will loop over the characters of each of the words.

```
words= ["Apple", "Banana", "Car", "Dolphin" ]
for word in words:
        #This loop is fetching word from the list
        print ("The following lines will print each letters of "+word)
        for letter in word:
                #This loop is fetching letter for the word
                print (letter)
        print("") #This print is used to print a blank line
```

For ... Loop in Python

❑ Break statement with for loop:

- The break statement is used to exit the for loop prematurely. It's used to break the for loop when a specific condition is met.
- Let's say we have a list of numbers and we want to check if a number is present or not. We can iterate over the list of numbers and if the number is found, break out of the loop because we don't need to keep iterating over the remaining elements.
- In this case, we'll use the Python if else condition along with our for loop.

```
nums = [1, 2, 3, 4, 5, 6]

n = 2

found = False
for num in nums:
    if n == num:
        found = True
        break

print(f'List contains {n}: {found}')

# Output
# List contains 2: True
```

For ... Loop in Python

❑ The continue statement with for loop:

- We can use continue statements inside a for loop to skip the execution of the for loop body for a specific condition.
- Let's say we have a list of numbers and we want to print the sum of positive numbers. We can use the continue statements to skip the for loop for negative numbers.

```
nums = [1, 2, -3, 4, -5, 6]

sum_positives = 0

for num in nums:
    if num < 0:
        continue
    sum_positives += num

print(f'Sum of Positive Numbers: {sum_positives}')
```

For ... Loop in Python

❑ Pass Statement in Python:

- The pass statement to write empty loops. Pass is also used for empty control statements, functions, and classes.

Python3

```
# An empty loop
for letter in 'geeksforgeeks':
    pass
print('Last Letter :', letter)
```

Output:

```
Last Letter : s
```

For ... Loop in Python

❑ Python for loop with an else block:

- We can use else block with a **Python for loop**. The else block is executed only when the **for loop** is not terminated by a break statement.
- Let's say we have a function to print the sum of numbers if and only if all the numbers are even.
- We can use break statement to terminate the for loop if an odd number is present. We can print the sum in the else part so that it gets printed only when the for loop is executed normally.

```
def print_sum_even_nums(even_nums):
    total = 0

    for x in even_nums:
        if x % 2 != 0:
            break

        total += x
    else:
        print("For loop executed normally")
        print(f'Sum of numbers {total}')


# this will print the sum
print_sum_even_nums([2, 4, 6, 8])

# this won't print the sum because of an odd number in the sequence
print_sum_even_nums([2, 4, 5, 8])

# Output

# For loop executed normally
# Sum of numbers 20
```

For ... Loop in Python

❑ Python for loop with an else block:

- Example:
- Break the loop when x is 3, and see what happens with the else block

```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")
```

For ... Loop in Python

❑ Python for loop with range() function:

Python range() is one of the built-in functions. When you want the for loop to run for a specific number of times, or you need to specify a range of objects to print out, the range function works really well. Consider the following example where I want to print the numbers 1, 2, and 3.

```
for x in range(3):
    print("Printing:", x)

# Output

# Printing: 0
# Printing: 1
# Printing: 2
```

The range function also takes another parameter apart from the start and the stop. This is the **step parameter**. It tells the range function how many numbers to skip between each count.

```
for n in range(1, 10, 3):
    print("Printing with step:", n)

# Output

# Printing with step: 1
# Printing with step: 4
# Printing with step: 7
```

In this example, I've used number 3 as the step and you can see the output numbers are the previous number + 3.

start: integer starting from which the sequence of integers is to be returned

stop: integer before which the sequence of integers is to be returned. The range of integers end at stop – 1.

step: integer value which determines the increment between each integer in the sequence

For ... Loop in Python

- **Python for loop with range() function:**

Example: performing sum of first 10 numbers

Python3

```
# Python Program to
# show range() basics

# printing a number
for i in range(10):
    print(i, end=" ")

# performing sum of first 10 numbers
sum = 0
for i in range(1, 10):
    sum = sum + i
print("\nSum of first 10 numbers :", sum)
```

For ... Loop in Python Array

- **Python For loop with array:**

A basic example for understanding how the numpy for loop works.

```
import numpy as np
arr1 = np.array([2, 1, 4])
for x in arr1:
print(x)
```

```
import numpy as np
arr1 = np.array([[2, 1, 4],[2, 4, 6]])
arr2 = np.array([[8, 16, 44],[22, 40, 16]])
arr3 = np.array([[7, 14, 21],[0, 4, 7]])
for x in arr1,arr2, arr3:
print(x)
```

Iterate through a two-dimensional array

```
import numpy as np
arr1 = np.array([[8, 16, 44],[22, 40, 16]])
for x in arr1:
for y in x:
print(y)
```

```
import numpy as np
arr = np.array([[[8, 16, 44], [0, 4, 7]], [[22, 40, 16], [7, 14, 21]]])
for x in arr:
for y in x:
for z in y:
print(z)
```

For ... Loop in Python Array

- **Python For loop with array:**

Looking at how to use for loops with numpy arrays, creating some arrays of random numbers.

```
import numpy as np
np.random.seed(0)  # seed for reproducibility
x = np.random.randint(10, size=6)
y = np.random.randint(10, size=6)
```

```
for val in x:
    print(val)
```

```
# creating our 2-dimensional array
z = np.array([x, y])

for val in z:
    print(val)
```

For ... Loop in Python Array

- **Python For loop with array:**

A different method that uses a numpy function nditer() which is widely used to loop through an array of different dimensions.

```
import numpy as np
arr = np.array([[[4, 3], [1, 4]], [[7, 6], [3, 2]]])
for x in np.nditer(arr):
print(x)
```

A three-dimensional 3-D array, and unlike python, for loop, we have iterated only once through each of the scalar values of the array.

For ... Loop in Python Array

- **Python For loop with Array and Pandas “DataFrames”:**

Working with a small CSV file that records the GDP, capital city, and population for six different countries. We will read this into a pandas DataFrame below.

Pandas works a bit differently from numpy, so we won’t be able to simply repeat the numpy process we’ve already learned. If we try to iterate over a pandas DataFrame as we would a numpy array.

```
import pandas as pd
df = pd.read_csv('gdp.csv', index_col=0)

for val in df:
    print(val)
```

```
Capital
GDP ($US Trillion)
Population
```

For ... Loop in Python Array

- **Python For loop with array:**

We need to mention explicitly that we want to iterate over the rows of the DataFrame. We do this by calling the iterrows() method on the DataFrame, and print row labels and row data, where a row is the entire pandas series.

```
for label, row in df.iterrows():
    print(label)
    print(row)
```

```
Ireland
Capital                 Dublin
GDP ($US Trillion)      0.3337
Population              4784000
Name: Ireland, dtype: object
United Kingdom
Capital                  London
GDP ($US Trillion)        2.622
Population              66040000
Name: United Kingdom, dtype: object
United States
Capital                 Washington, D.C.
GDP ($US Trillion)                 19.39
Population                     327200000
```

We can also access specific values from a pandas series.

```
for label, row in df.iterrows():
    print(label + " : " + row["Capital"])
```

```
Ireland : Dublin
United Kingdom : London
United States : Washington, D.C.
China : Beijing
India : New Delhi
Germany : Berlin
```

For ... Loop in Python Array

- **Python For loop with array:**

To take things further than simple printouts, let's add a column using a for loop. Let's add a GDP per capita column. Remember that .loc[] is label-based.

We'll add the column and compute its contents for each country by dividing its total GDP from its population and multiplying the result by one trillion (since the GDP numbers are listed in trillions).

```
for label, row in df.iterrows():
    df.loc[label,'gdp_per_cap'] = row['GDP ($US Trillion)']/row['Population '] * 1000000000000
print(df)
```

```
                      Capital  GDP ($US Trillion)  Population   \
Country
Ireland                Dublin              0.3337      4784000
United Kingdom         London              2.6220     66040000
United States   Washington, D.C.             19.3900    327200000
China                   Beijing             12.2400   1386000000
India                 New Delhi              2.5970   1339000000
Germany                  Berlin              3.6770     82790000
```

For ... Loop in Python Array

A story of Fibonacci Series

For ... Loop in Python Array – A story of Fibonacci Series

- **Fibonacci Series in Python using For Loop:**

The Fibonacci sequence was developed by the Italian mathematician, Leonardo Fibonacci, in the 13th century. The sequence of numbers, starting with zero and one, is a steadily increasing series where each number is equal to the sum of the preceding two numbers.

Fibonacci Sequence Rule

$xn = xn-1 + xn-2$

where:

xn is term number "n"

xn−1 is the previous term (n−1)

xn−2 is the term before that (n−2)

Fibonacci Series

Default

0 1 1 2 3 5

0 + 1 = 1

1 + 1 = 2

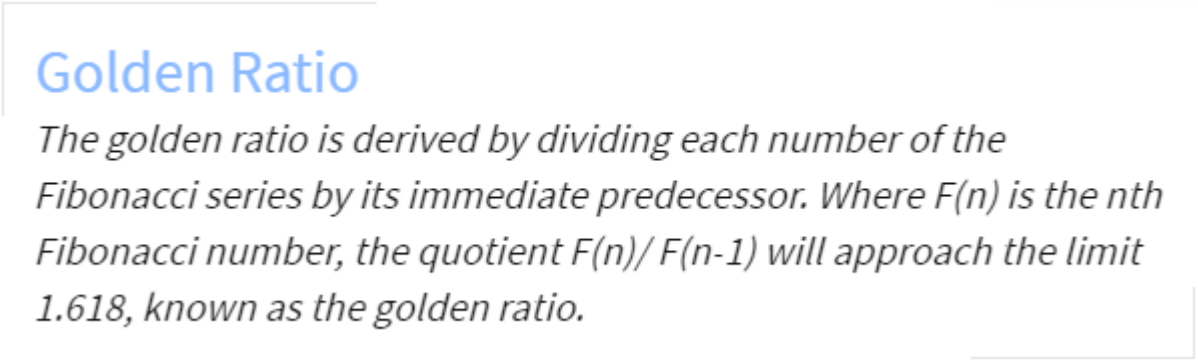
1 + 2 = 3

2 + 3 = 5

For ... Loop in Python Array

- **Fibonacci Series in Python using For Loop:**

- Fibonacci sequence of numbers and the associated “Golden Ratio” are manifested in nature and in certain works of art
- Some traders believe that the Fibonacci numbers and ratios created by the sequence play an important role in finance that traders can apply using technical analysis.

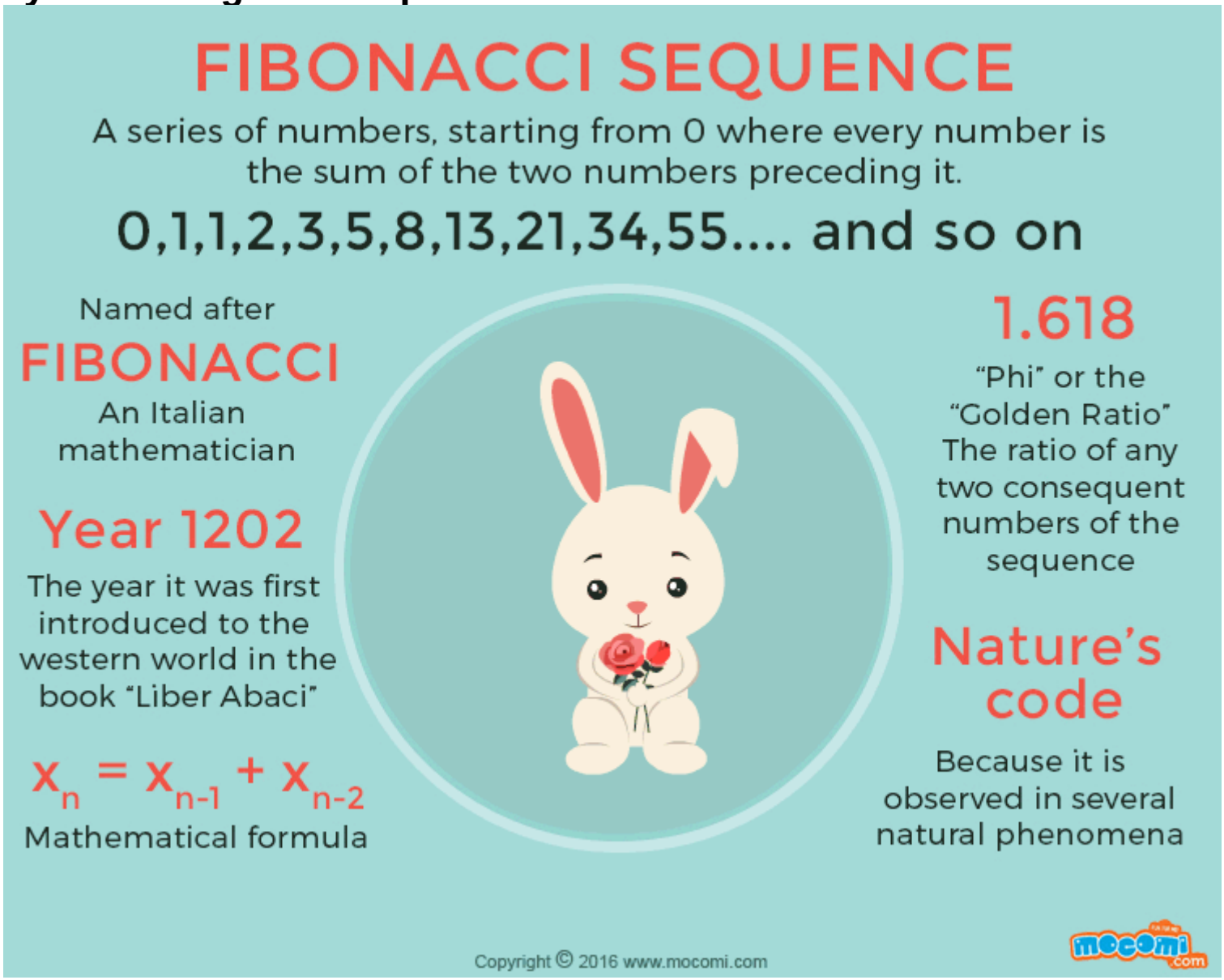


The GOLDEN ratio of 1.618, important to mathematicians, scientists, and naturalists for centuries is derived from the Fibonacci sequence.

The quotient between each successive pair of Fibonacci numbers in the sequence approximates 1.618, or its inverse 0.618.

For ... Loop in Python Array

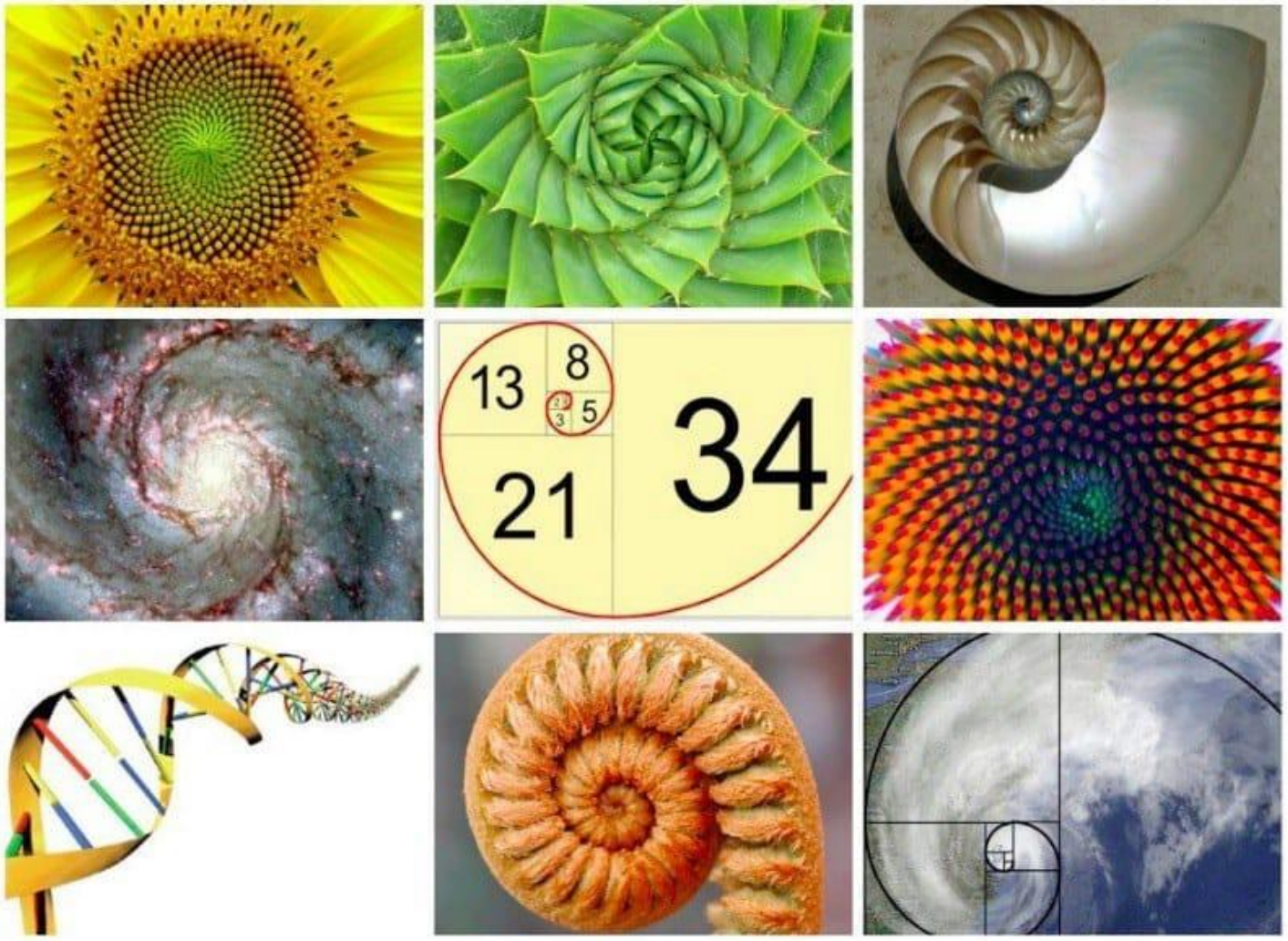
- **Fibonacci Series in Python using For Loop:**



For ... Loop in Python Array

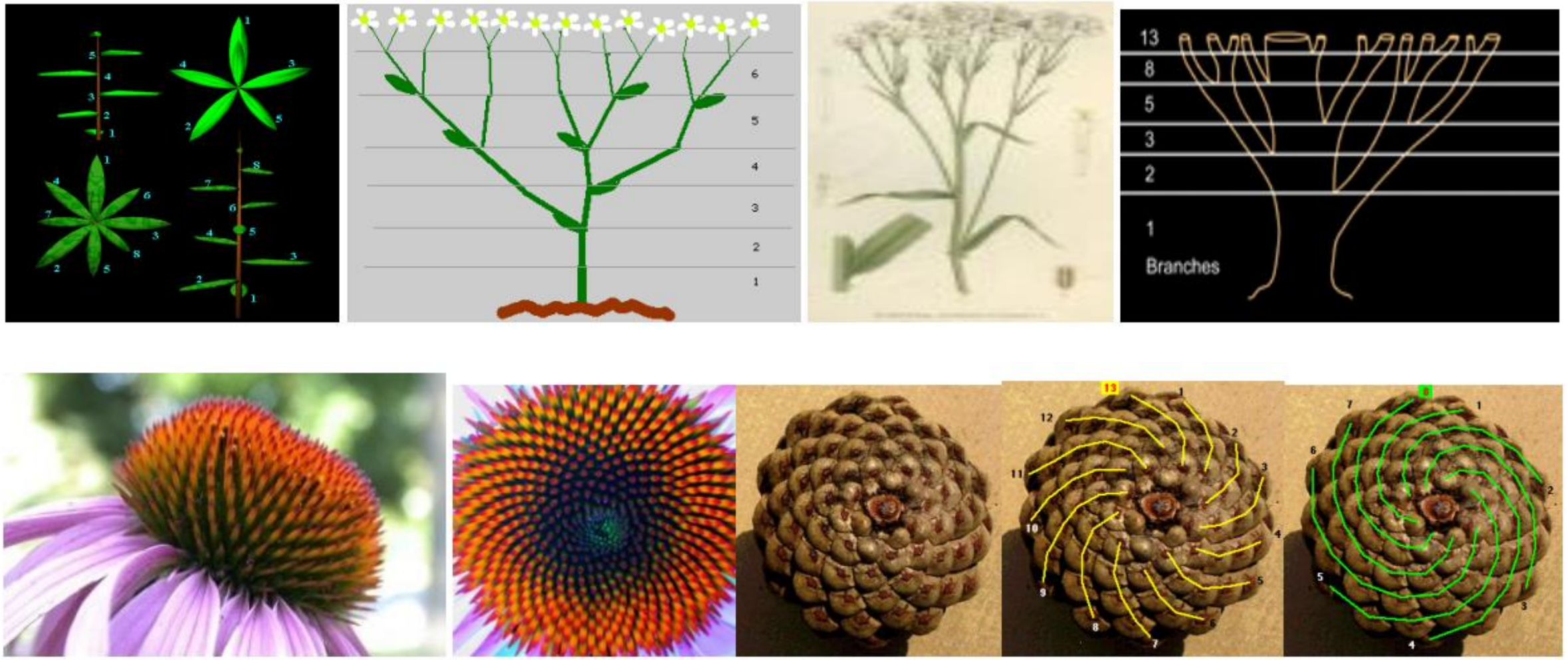
- **Fibonacci Series in Python using For Loop:**

The Fibonacci sequence was developed by the Italian mathematician, Leonardo Fibonacci, in the 13th century. The sequence of numbers, starting with zero and one, is a steadily increasing series where each number is equal to the sum of the preceding two numbers.



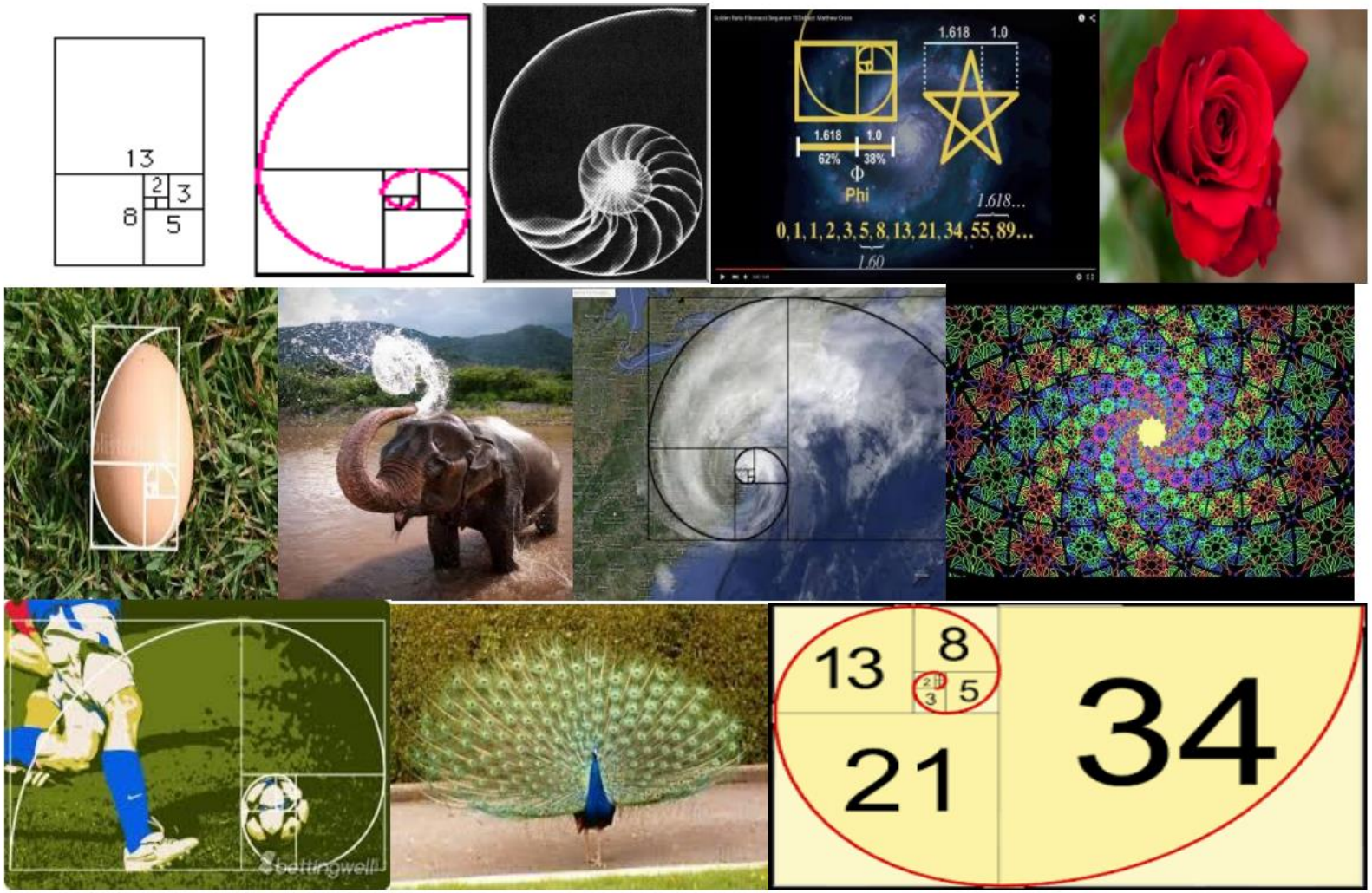
For ... Loop in Python Array

- **Fibonacci Series in Python using For Loop:**



For ... Loop in Python Array

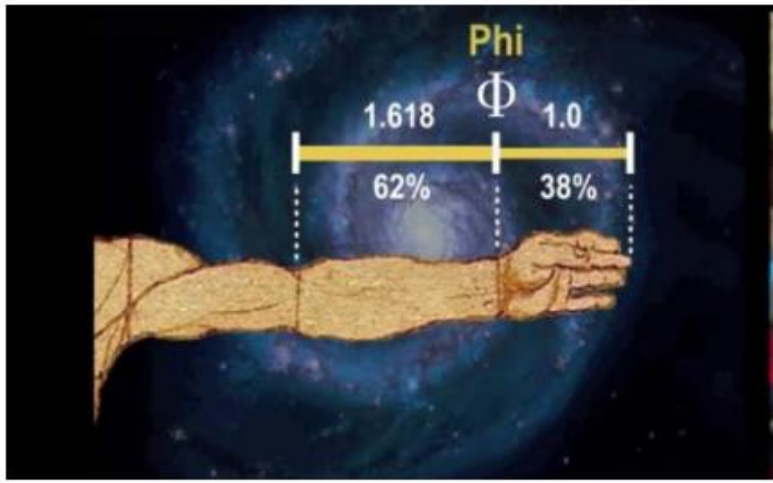
- **Fibonacci Series in Python using For Loop:**

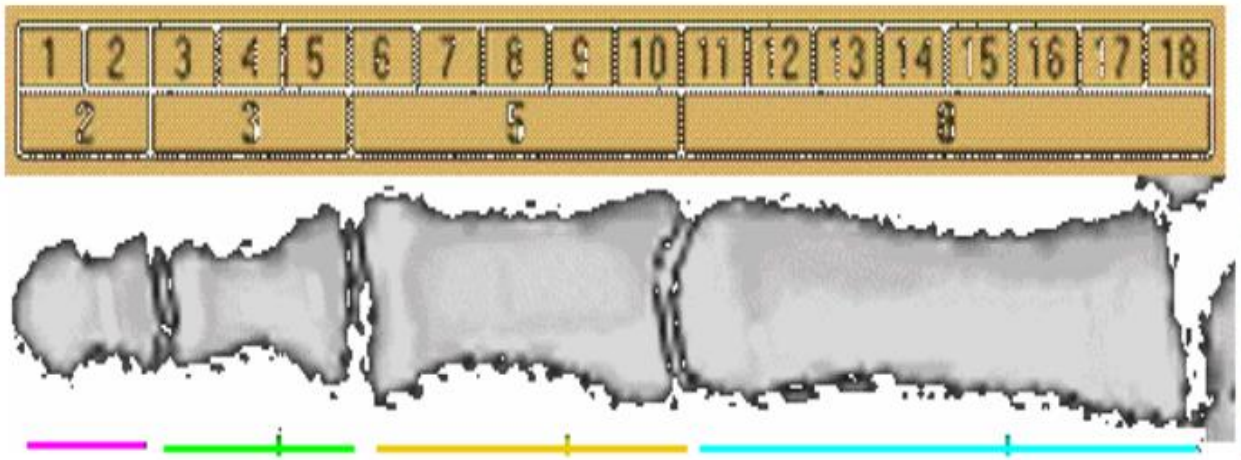


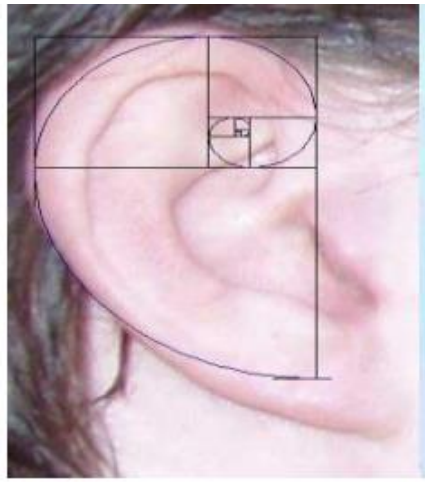
For ... Loop in Python Array

- **Fibonacci Series in Python using For Loop:**

The Fibonacci Numbers and Its Amazing Applications



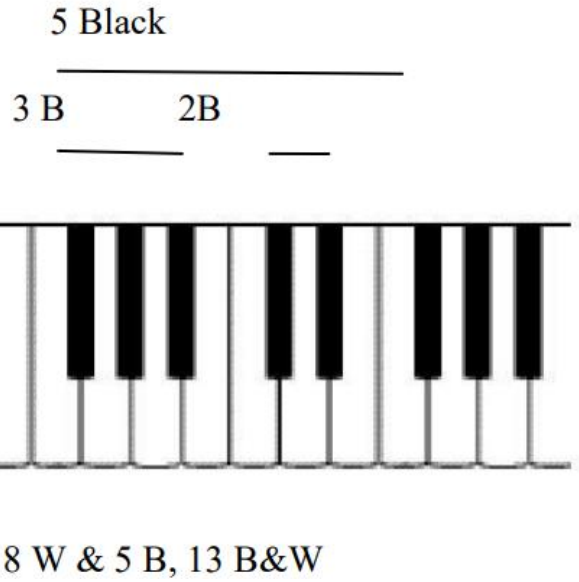


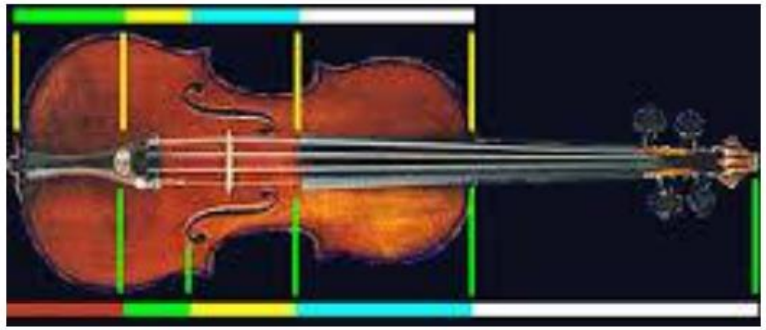


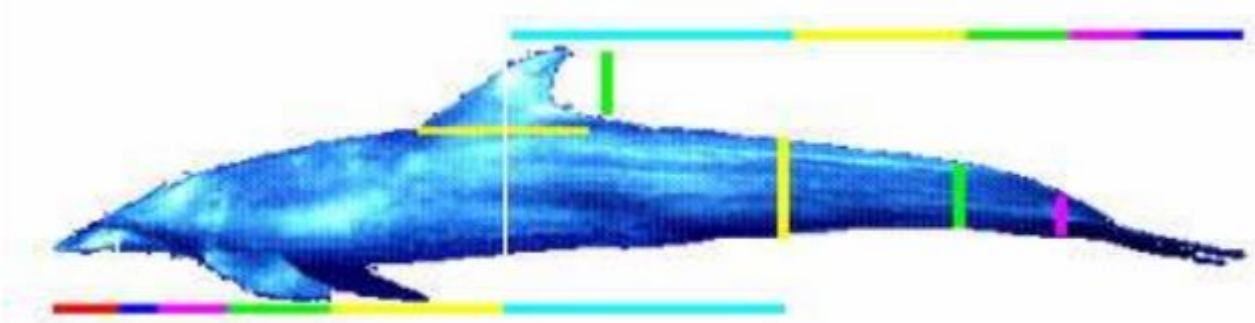
For ... Loop in Python Array

- **Fibonacci Series in Python using For Loop:**

The Fibonacci Numbers and Its Amazing Applications





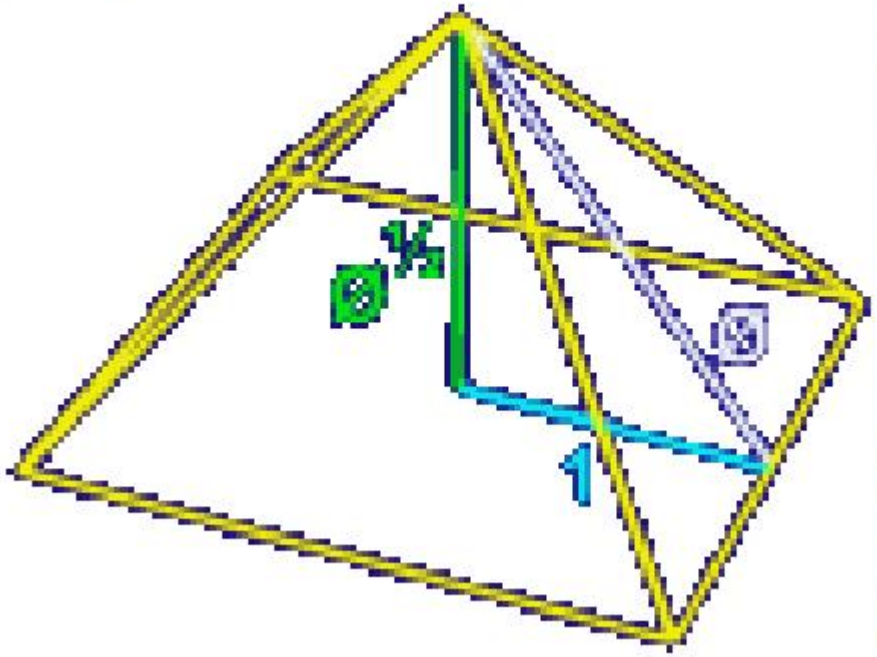


For ... Loop in Python Array

- **Fibonacci Series in Python using For Loop:**

The Fibonacci Numbers and Its Amazing Applications

The Golden Ratio is also frequently seen in natural architecture also (Internet access, 18). It can be found in the great pyramid in Egypt. Perimeter of the pyramid, divided by twice its vertical height is the value of ϕ .



For ... Loop in Python Array

Fibonacci Series in Python using For Loop:

- We read a number from user, N as input. N represents the number of elements of Fibonacci Series to be generated and print to the console.

```
N = int(input("Number of elements in Fibonacci Series, N, (N>=2) : "))

#initialize the list with starting elements: 0, 1
fibonacciSeries = [0,1]

if N>2:
    for i in range(2, N):
        #next elment in series = sum of its previous two numbers
        nextElement = fibonacciSeries[i-1] + fibonacciSeries[i-2]
        #append the element to the series
        fibonacciSeries.append(nextElement)

print(fibonacciSeries)
```

Bài tập thực hành

Start/Stop

Input/Output

Do something

Decision

Tạo một ứng dụng, hiển thị chuỗi Fibonacci theo kích thước yêu cầu.

1. Viết function thực hiện tính toán và hiển thị chuỗi Fibonacci theo đúng chuẩn.
2. Viết một chương trình "my_program" dạng login (với password) thực hiện nhiệm vụ sau:

   Tạo một vòng lặp vô tận và thao tác login input với password={DH21HM}
   để vào chương trình:

   a. Nếu nhập đúng mã thì thực hiện function Fibonacci_Series, trong đó tiếp tục tạo input để yêu cầu nhập kích thước của chuỗi Fibonacci và hiển thị chuỗi sau khi tính toán.

      i. Sau khi thực hiện xong function Fibonacci, tạo input yêu cầu thoát chương trình hay không, nếu có nhập mã {Fibonacci} để thoát ra và thực hiện bước code tiếp theo.

      ii. Nếu không, tạo delay chờ 3s, và tiếp tục vòng lặp vô tận ban đầu.

   b. Nếu nhập sai mật mã, thì không thực hiện function Fibonacci_Series và yêu cầu nhập lại. Ngoài ra, viết một thao tác vòng lặp, sau 3 lần nhập mã không đúng thì sẽ thoát hẳn chương trình.

3. Sau khi thoát ra "my_program", sử dụng chuỗi Fibonacci_Series đã tìm được plot chuỗi Fibonacci này với các giá trị của chuỗi.