

# LAB 1: LÀM QUEN VỚI NGÔN NGỮ LẬP TRÌNH PYTHON

(SV thực hiện tại lớp)

\*\*\*

## Bài tập 1: Cài đặt chương trình Python và thư viện có sẵn trong bộ Anaconda

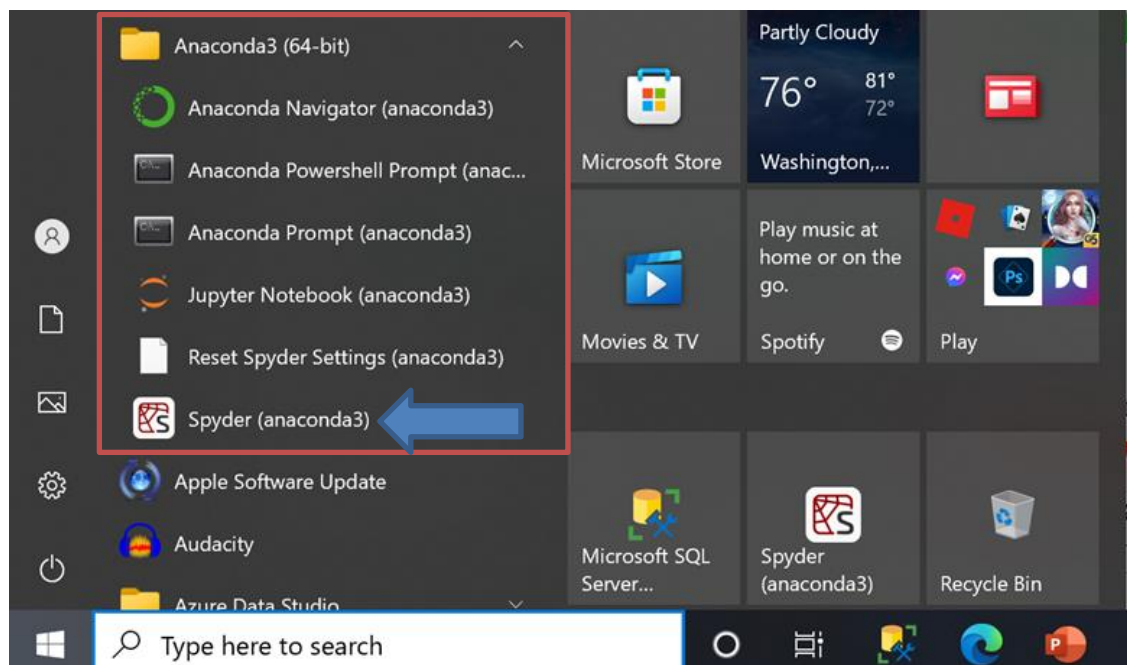
### Cài đặt Anaconda

- **Anaconda** là một bản phân phối các ngôn ngữ lập trình Python và R cho tính toán khoa học, nhằm mục đích đơn giản hóa việc triển khai và quản lý gói. Bản phân phối bao gồm các gói khoa học dữ liệu phù hợp với **Windows**, **Linux** và **macOS**.

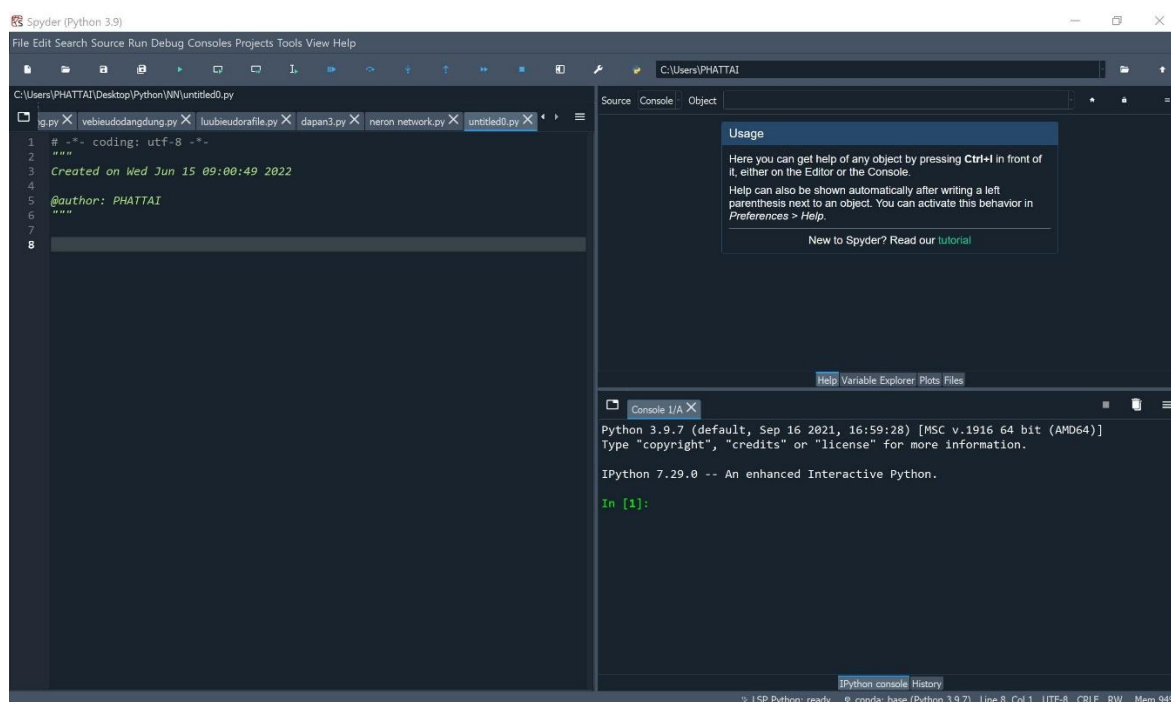
#### Hướng dẫn:

- Download gói cài đặt tại: Phiên bản hiện tại là: **Anaconda3-2022.05**

[https://repo.anaconda.com/archive/Anaconda3-2022.05-Windows-x86\\_64.exe](https://repo.anaconda.com/archive/Anaconda3-2022.05-Windows-x86_64.exe)



- Trong bộ Anaconda3(64 bit) sử dụng chương trình Spyder để editor code Python



## Bài tập 2: Thao tác làm việc trong Python

### Tìm hiểu về các thư viện trong Python

#### 2.1. Thư viện NumPy

- **NumPy** là một gói xử lý (Processing Package) phổ biến của **Python**. **NumPy** làm phong phú ngôn ngữ lập trình Python với các cấu trúc dữ liệu mạnh mẽ để tính toán hiệu quả các mảng và ma trận đa chiều. **NumPy** không chỉ là một gói module để xử lý mảng mà nó còn cung cấp khả năng quản lý mảng cực kỳ vượt trội.

#### 2.2. Thư viện SciPy

- Thư viện hữu ích này bao gồm các module cho đại số tuyến tính, tích hợp, tối ưu hóa và thống kê. Chức năng chính của nó được xây dựng dựa trên NumPy, vì vậy các mảng của nó sử dụng thư viện này. SciPy hoạt động hiệu quả cho tất cả các loại dự án lập trình khoa học (khoa học, toán học và kỹ thuật). Nó cung cấp các quy trình số hiệu quả như tối ưu hóa số, tích hợp và các quy trình khác trong module con. Tài liệu phong phú giúp làm việc với thư viện này thực sự dễ dàng.

#### 2.3. Thư viện Pandas

- **Pandas** là một thư viện được tạo ra để giúp các nhà phát triển làm việc với dữ liệu "labeled" và "relational" một cách trực quan. Nó dựa trên hai cấu trúc dữ liệu chính: "Chuỗi" (một chiều, giống như danh sách các mục) và "Khung dữ liệu" (hai chiều, giống như một bảng có nhiều cột). Pandas cho phép chuyển đổi cấu trúc dữ liệu thành các đối tượng DataFrame, xử lý dữ liệu bị thiếu và thêm / xóa các cột khỏi DataFrame, đưa vào các tệp bị thiếu và vẽ dữ liệu bằng biểu đồ hoặc hộp biểu đồ. Đây là điều bắt buộc phải có để xử lý dữ liệu, thao tác và trực quan hóa.

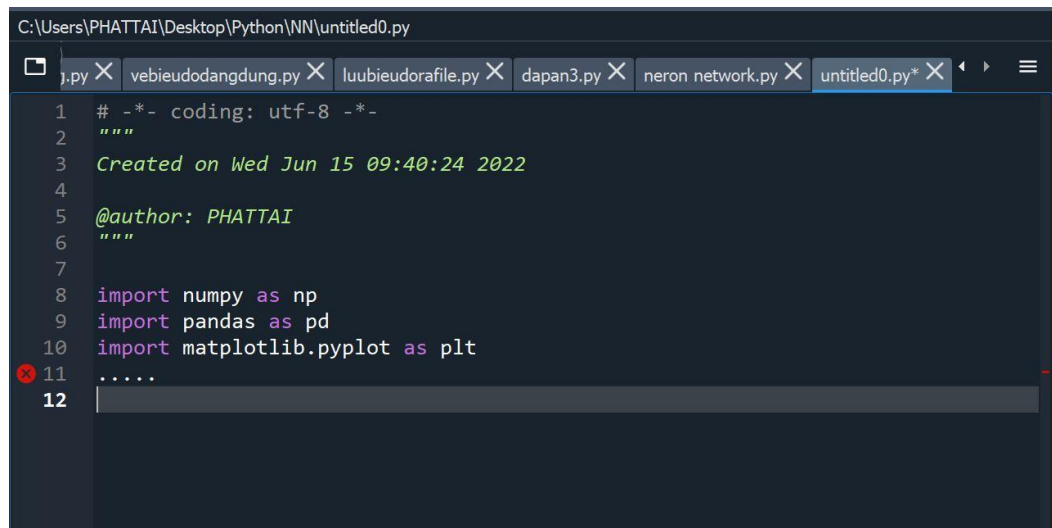
#### 2.4. Thư viện Matplotlib

- Đây là một thư viện khoa học dữ liệu tiêu chuẩn giúp tạo ra các trực quan hóa dữ liệu như biểu đồ và biểu đồ hai chiều (biểu đồ, biểu đồ phân tán, biểu đồ tọa độ phi Descartes). Matplotlib là một trong những thư viện vẽ biểu đồ thực sự hữu ích trong các dự án khoa học dữ liệu - nó cung cấp một API hướng đối tượng để nhúng các biểu đồ vào ứng dụng.

## 2.5. Thư viện Keras

- **Keras** là một thư viện tuyệt vời để xây dựng mạng nơ-ron và mô hình hóa. Nó rất dễ sử dụng và cung cấp cho các nhà phát triển một mức độ mở rộng tốt. Thư viện tận dụng các gói khác (**Theano** hoặc **TensorFlow**) làm phụ trợ của nó.

**Cấu trúc chung:**



```
C:\Users\PHATTAI\Desktop\Python\NN\untitled0.py
untitled0.py x vebieudodangdung.py x luubieudorafale.py x dapan3.py x neron network.py x untitled0.py* x
1  # -*- coding: utf-8 -*-
2  """
3  Created on Wed Jun 15 09:40:24 2022
4
5  @author: PHATTAI
6  """
7
8  import numpy as np
9  import pandas as pd
10 import matplotlib.pyplot as plt
11 .....
12 |
```

## Bài tập 3: Ôn tập biến và hàm trong Python

### 3.1. Khai báo biến trong Python.

```
tenBien = giaTri
```

+ **tenBien** là tên của biến mà các bạn muốn đặt. Tên biến này không được bắt đầu bằng số hay các ký tự đặc biệt, mà chỉ được bắt đầu bằng chữ cái hoặc ký tự `_` và nó có phân biệt hoa thường.

+ **giaTri** là giá trị của biến mà bạn muốn gán.

### 3.2. Các kiểu dữ liệu trong Python

```
name = "Vũ Thanh Tài"
#string

age = 22
#integer

point = 8.9
#float

option = [1,2,3,4,5]
#lists

tuple = ('Vũ Thanh Tài', 22 , True)
#Tuple

dictionary = {"name": "Vu Thanh Tai", "age": 22, "male": True}
#Dictionary
```

### 3.3.Các kiểu dữ liệu trong Python

Trong một trường hợp nào đó mà bạn muốn chuyển đổi kiểu dữ liệu của một biến, thì Python cũng hỗ trợ bạn qua các hàm cơ bản sau:

- *float(data)* chuyển đổi sang kiểu số thực.
- *int(data,base)* chuyển đổi sang kiểu số, trong đó base là kiểu hệ số mà các bạn muốn chuyển đổi sang (tham số này có thể bỏ trống).
- *str(data)* chuyển đổi sang dạng chuỗi.
- *complex(data)* chuyển đổi sang kiểu phức hợp.
- *tuple(data)* chuyển đổi sang kiểu Tuple.
- *dict(data)* chuyển đổi sang kiểu Dictionary.
- *hex(data)* chuyển đổi sang hệ 16.
- *oct(data)* chuyển đổi sang hệ 8.
- *chr(data)* chuyển đổi sang dạng ký tự.
- .....

**Ví dụ:**

```
age = 22;

# ép sang float
floatAge = float(age)
print(type(floatAge))

#ép sang integer.
intAge = int(age)
print(type(intAge))

#ép sang chuỗi.
strAge = str(age)
print(type(strAge))
```

### 3.4. Hàm trong Python

Khi muốn sử dụng một hàm **Python** do người dùng định nghĩa, trước hết chúng ta cần khai báo hàm trong **python** bằng cách sử dụng **def** trong **python** với cú pháp hàm như sau:

```
def  tên hàm ( tham số 1 ,  tham số 2 ,  ... ) :
    Câu lệnh 1 trong hàm
    Câu lệnh 2 trong hàm
    ...
    return  giá trị trả về
```

- **def** là từ khóa dùng để khai báo hàm trong Python
- **tên hàm:** là một chuỗi ký tự dùng để đặt tên đại diện cho hàm.
- **tham số:** là các biến sử dụng trong khai báo hàm, cũng như để nhập đối số khi chúng ta sử dụng hàm.
- **return:** là từ khóa dùng để trả giá trị trả về từ hàm trong Python

**Ví dụ:** Sử dụng **def** để tạo ra một hàm tính tổng hai số và trả về kết quả trong chương trình như sau:

```
def add(a, b):  
    x = a + b  
    return x
```

Nếu một hàm không cần nhận giá trị truyền vào từ bên ngoài thì chúng ta cũng có thể lược bỏ tham số khi khai báo hàm trong python như sau:

```
def tên_hàm () :  
    Câu_lệnh_1_trong_hàm  
    Câu_lệnh_2_trong_hàm  
    ...  
    return giá_trị_trả_về
```

**Ví dụ:** Chúng ta không sử dụng tham số khi khai báo hàm:

```
def add():  
    x = 1 + 2  
    print(x)  
    return x
```

Sau khi khai báo hàm trong **python**, chúng ta có thể sử dụng hàm đó nhiều lần trong chương trình bằng cách gọi **hàm trong python** với cú pháp sau đây:

```
tên_hàm ( đối_số_1 , đối_số_2 , ... )
```

**Ví dụ:** Chúng ta khai báo và gọi hàm trong python như sau::

```
def add(a, b):  
    x = a + b  
    return(x)  
  
add(1, 2)  
add(5, 6)
```

### 3.5. Câu lệnh điều khiển

Khối câu lệnh được phân định bằng khoảng trống (spaces hay tabs) ở đầu các dòng lệnh. Đây chính là sự khác biệt của Python với các ngôn ngữ khác.

```

var1 = 10
var2 = 20
if var1 != var2:
    print("var1 is not equal to var2")
elif var1 > var2:
    print("var1 is greater than var2")
elif var2 > var1:
    print("var2 is greater than var1")
else:
    print("var1 is equal to var2")

```

### 3.6. Vòng lặp While

```

count = 0
fruits = ['banana', 'apple', 'mango', 'melon', 'orange', 'pear', 'pineapple']
while count < len(fruits):
    print('Current fruit:', fruits[count])
    count += 1

```

### 3.7. Vòng lặp For

```

fruits = ['banana', 'apple', 'mango', 'melon', 'orange', 'pear', 'pineapple']
for index in range(len(fruits)):
    print('Current fruit:', fruits[index])

```

## Bài tập 4: Thực hiện một số bài toán trong Python

- 4.1. Viết chương trình để tạo mảng một chiều giá trị gồm 10 phần tử được sinh số ngẫu nhiên trong khoảng từ [0:1]

🔧 **Gợi ý**

Phương thức **randn()** ngẫu nhiên **numpy** chỉ nhận một chiều và trả về mảng một chiều.

- 4.2. Viết chương trình thông qua khai báo hàm **Tong(w1,a1,w2,a2,b)** để tính tổng công thức sau:

$$y = (w1 * a1 + w2 * a2) + bias$$

Trong đó:

**w1,w2,bias** = ramdon số trong khoảng [0:1]

**a1,a2**: Lần lượt thay bằng các giá trị trong bảng:

<b>a1</b>	3	2	4	3	3.5
<b>a2</b>	1.5	1	1.5	1	0.5

- 4.3. Viết chương trình tạo ra 1 mảng hai chiều 4x5 giá trị gồm các số ngẫu nhiên trong khoảng từ [0:1]

🔧 **Gợi ý**: `data = np.random.randn(4, 5)`



**4.4. Viết chương trình tạo Game đoán số như sau:**

- Máy tính sẽ random các số chạy từ 1 – 50
- Người chơi sẽ có 3 lần để đoán trúng. Nếu đoán không trúng, người chơi sẽ được gợi ý số vừa đoán nhỏ hơn hay lớn hơn kết quả. Đoán 3 lần mà không trúng thì coi như thua.

🔧 Gợi ý:

```
import random

so_lan_traLoi = 0
so_random = random.randint(1, 50)

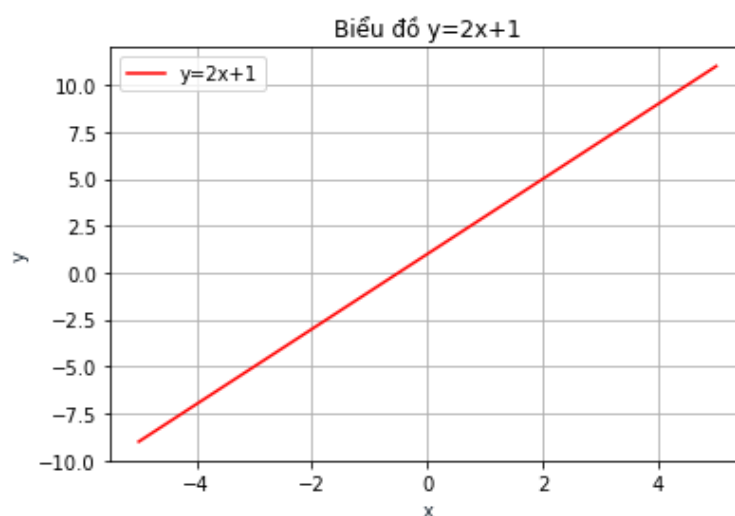
# print(so_random)

print("Mình đang nghĩ một con số từ 1 - 50 \r")

while so_lan_traLoi < 3:
    so_lan_traLoi += 1
    doan = input("Con số dự đoán của bạn lần thứ "+ str(so_lan_traLoi)+ " là: ")
    doan = int(doan)

    if doan < so_random:
        print("Bạn đoán nhỏ hơn rồi nhé ")
    elif doan > so_random:
        print("Bạn đoán Lớn hơn rồi nhé ")
    else:
        break

if doan == so_random:
    print("Bạn đoán đúng rồi, chúc mừng bạn, con số dễ đoán phải ko ??")
else:
    print("Sau tất cả, hẹn gặp lại bạn lần sau !!!, con số mình nghĩ là " + str(so_ran
```

**4.5. Viết chương trình vẽ biểu đồ  $y = 2x+1$  theo hình sau:**



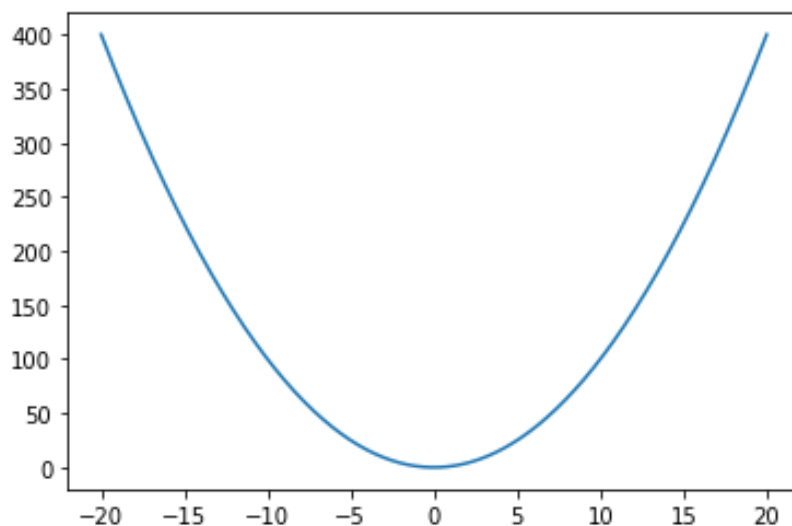
🚦 Gợi ý:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 15 10:07:50 2022

@author: PHATTAI
"""

import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-5,5,100)
y = 2*x+1
plt.plot(x, y, '-r', label='y=2x+1')
plt.title('Biểu đồ y=2x+1')
plt.xlabel('x', color='#1C2833')
plt.ylabel('y', color='#1C2833')
plt.legend(loc='upper left')
plt.grid()
plt.show()
```

4.6. Viết chương trình vẽ biểu đồ  $y = x^2$  theo hình sau:



🚦 Gợi ý:

```
1 import numpy as np # thư viện numpy
2 import matplotlib.pyplot as plt # thư viện pyplot
3 # chia đoạn từ -20 đến 20 thành 1000 đoạn
4 x = np.linspace(-20, 20, 1000)
5 # tính y
6 y = x * x
7 # vẽ biểu đồ tương quan giữa x và y
8 plt.plot(x, y)
9 # hiển thị biểu đồ
10 plt.show()
11
```