# Giới Thiệu Về Python

## 1. Lịch Sử Hình Thành Và Phát Triển Của Python

Python được Guido van Rossum giới thiệu lần đầu vào năm 1991. Ban đầu, Python được thiết kế để dễ học và thân thiện với người dùng, tập trung vào tính dễ đọc. Qua thời gian, Python đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất, được sử dụng trong nhiều lĩnh vực như khoa học dữ liệu, trí tuệ nhân tạo, và phát triển web.

## 2. Các Kiểu Dữ Liệu Trong Python

Python cung cấp nhiều kiểu dữ liệu khác nhau bao gồm:

- **Kiểu số**: `int`, `float`, `complex`
  Ví dụ:

```
x = 10       # int
y = 3.14     # float
z = 1 + 2j   # complex
print(type(x), type(y), type(z))
```

- **Kiểu chuỗi**: `str`
  Ví dụ:

```
name = "Python"
print(name, type(name))
```

- **Kiểu danh sách**: `list`
  Ví dụ:

```
fruits = ["apple", "banana", "cherry"]
print(fruits, type(fruits))
```

- **Kiểu tuple**: `tuple`
  Ví dụ:

```
coordinates = (10, 20, 30)
print(coordinates, type(coordinates))
```

- **Kiểu tập hợp**: `set`, `frozenset`
  Ví dụ:

```
unique_numbers = {1, 2, 3, 4}
print(unique_numbers, type(unique_numbers))
```

- **Kiểu từ điển**: `dict`
  Ví dụ:

```
student = {"name": "John", "age": 20}
print(student, type(student))
```

- **Kiểu logic**: `bool`
  Ví dụ:

```
is_active = True
print(is_active, type(is_active))
```

- **Kiểu rỗng**: `NoneType`
  Ví dụ:

```
data = None
print(data, type(data))
```

### 3. Cách Khai Báo Biến

Trong Python, biến được khai báo bằng cách gán giá trị trực tiếp:

```
x = 10  # Biến nguyên
name = "Python"  # Biến chuỗi
is_active = True  # Biến logic
```

### 4. Các Phép Tính Và Toán Tử Cơ Bản

Python cung cấp nhiều toán tử cơ bản:

- Phép cộng: `+`
  Ví dụ:

```
a = 5
b = 3
print(a + b)  # Kết quả: 8
```

- Phép trừ: `-`
  Ví dụ:

```
a = 5
b = 3
print(a - b)  # Kết quả: 2
```

- Phép nhân: `*`
  Ví dụ:

```
a = 5
b = 3
print(a * b)  # Kết quả: 15
```

- Phép chia: `/`
  Ví dụ:

```
a = 5
b = 2
print(a / b)  # Kết quả: 2.5
```

- Phép chia nguyên: `//`
  Ví dụ:

```
a = 5
b = 2
print(a // b)  # Kết quả: 2
```

- Phép chia lấy dư: `%`
  Ví dụ:

```
a = 5
b = 2
print(a % b)  # Kết quả: 1
```

- Phép luỹ thừa: `**`
  Ví dụ:

```
a = 2
b = 3
print(a ** b)  # Kết quả: 8
```

## 5. Các Câu Điều Kiện Và Vòng Lặp

### Câu Điều Kiện

Sử dụng `if`, `elif`, `else`:

```
x = 10
if x > 5:
    print("x lớn hơn 5")
elif x == 5:
    print("x bằng 5")
else:
    print("x nhỏ hơn 5")
```

### Vòng Lặp

- **Vòng lặp `for`:**

```
for i in range(5):
    print(i)
```

- **Vòng lặp `while`:**

```
x = 0
while x < 5:
    print(x)
    x += 1
```

## 6. Ví Dụ: Giải Phương Trình Bậc Hai

Phương trình bậc hai có dạng:

$ax^2 + bx + c = 0$

Code Python:

```
import math

def solve_quadratic(a, b, c):
    # Tính địa
    delta = b**2 - 4*a*c
    if delta < 0:
        return "Phương trình vô nghiệm"
    elif delta == 0:
        x = -b / (2*a)
        return f"Phương trình có nghiệm kép: x = {x}"
    else:
        x1 = (-b + math.sqrt(delta)) / (2*a)
        x2 = (-b - math.sqrt(delta)) / (2*a)
        return f"Phương trình có hai nghiệm: x1 = {x1}, x2 = {x2}"
```

```
# Ví dụ
print(solve_quadratic(1, -3, 2))
```

Khi chạy code trên, kết quả sẽ là:

```
Phương trình có hai nghiệm: x1 = 2.0, x2 = 1.0
```

## 7. List trong Python

`List` là một trong những cấu trúc dữ liệu quan trọng trong Python, cho phép lưu trữ nhiều giá trị trong một danh sách có thể thay đổi.

### Tạo danh sách

```
# Tạo danh sách
my_list = [1, 2, 3, 4, 5]
print(my_list)
```

### Truy cập phần tử

```
# Truy cập phần tử đầu tiên
print(my_list[0])

# Truy cập phần tử cuối cùng
print(my_list[-1])
```

### Thêm, xóa phần tử

```
# Thêm phần tử vào danh sách
my_list.append(6)  # Thêm vào cuối
my_list.insert(2, 10)  # Chèn vào vị trí index 2
print(my_list)

# Xóa phần tử
my_list.remove(10)  # Xóa giá trị 10
print(my_list)
```

### Duyệt danh sách

```
# Duyệt danh sách bằng vòng lặp for
for item in my_list:
    print(item)
```

### Danh sách hiểu (List Comprehension)

```
# Tạo danh sách mới từ danh sách cũ
squared_list = [x**2 for x in my_list]
print(squared_list)
```

## 8. Array trong Python

Trong Python, cấu trúc dữ liệu danh sách ( `list` ) có thể hoạt động như một mảng. Tuy nhiên, Python không có kiểu dữ liệu mảng riêng biệt mà sử dụng `list` hoặc module `array` để lưu trữ các giá trị cùng kiểu.

### Ví dụ với `list`:

```
# Tạo một danh sách (list) trong Python
arr = [1, 2, 3, 4, 5]
print(arr)
```

### Ví dụ với module `array`:

```
import array

# Tạo một mảng số nguyên
arr = array.array('i', [1, 2, 3, 4, 5])
print(arr)
```

Ở đây, `'i'` đại diện cho kiểu dữ liệu số nguyên (integer).

## 9. Array trong NumPy

NumPy là thư viện mạnh mẽ giúp làm việc với mảng hiệu quả hơn, hỗ trợ nhiều thao tác tính toán khoa học.

### Khai báo mảng NumPy

```
import numpy as np

# Tạo một mảng NumPy từ danh sách Python
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

### Các thao tác cơ bản với NumPy Array

```
# Tạo mảng 2 chiều
matrix = np.array([[1, 2, 3], [4, 5, 6]])
print(matrix)

# Lấy kích thước của mảng
print("Shape:", matrix.shape)

# Truy cập phần tử
print("Phần tử hàng 1, cột 2:", matrix[0, 1])

# Thực hiện phép toán
arr2 = arr * 2  # Nhân mỗi phần tử với 2
print(arr2)
```

### Các phép toán với ma trận NumPy

```
# Cộng hai ma trận
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
C = A + B
print("Ma trận tổng:\n", C)

# Nhân hai ma trận
D = np.dot(A, B)
print("Ma trận tích:\n", D)

# Tính ma trận chuyển vị
E = A.T
print("Ma trận chuyển vị:\n", E)

# Tính định thức
det_A = np.linalg.det(A)
print("Định thức của A:", det_A)

# Tính ma trận nghịch đảo
inv_A = np.linalg.inv(A)
print("Ma trận nghịch đảo của A:\n", inv_A)
```

## 10. Array trong Pandas

Pandas sử dụng cấu trúc dữ liệu `Series` và `DataFrame` để lưu trữ dữ liệu dạng bảng.

### Tạo `Series`

```
import pandas as pd

# Tạo Series từ danh sách
```

```
series = pd.Series([1, 2, 3, 4, 5])
print(series)
```

### Tạo `DataFrame`

```
# Tạo DataFrame từ danh sách
data = {
    'A': [1, 2, 3],
    'B': [4, 5, 6]
}
df = pd.DataFrame(data)
print(df)
```

### Truy cập dữ liệu trong DataFrame

```
# Truy cập cột 'A'
print(df['A'])

# Truy cập hàng đầu tiên
print(df.iloc[0])
```

### Đọc dữ liệu từ file CSV và Excel

```
# Đọc file CSV
csv_df = pd.read_csv("data.csv")
print(csv_df.head())

# Đọc file Excel
excel_df = pd.read_excel("data.xlsx", sheet_name="Sheet1")
print(excel_df.head())
```