# **Báo cáo Python**

## 1. Tổng quan về lập trình Python

1.1 Lịch sử phát triển của Python

* Năm 1980: Guido van Rossum bắt đầu làm việc trên Python tại Trung tâm nghiên cứu Math và Computer Science ở Amsterdam, sau khi làm việc trên dự án ABC.
* Năm 1989: Guido bắt đầu dự án mới, Python, với mục tiêu tạo ra một ngôn ngữ lập trình mạnh mẽ, dễ đọc và dễ hiểu.
* Năm 1991: Python 0.9.0 được phát hành vào tháng 2 và Python 1.0 được phát hành vào tháng 1 năm 1994, với nhiều tính năng mới và cải tiến.
* Năm 2000: Python 2.0 được phát hành với nhiều cải tiến so với các phiên bản trước đó.
* Năm 2008: Python 3.0 (Python 3000 hoặc Py3k) được phát hành, mang lại nhiều cải tiến lớn và làm sạch ngôn ngữ, nhưng cũng gây ra một số không tương thích với mã Python 2.x.
* Năm 2020: Python 2.x chính thức kết thúc vòng đời hỗ trợ vào ngày 1 tháng 1, khuyến khích người dùng chuyển sang Python 3.x.
* Năm 2023: Python 3.12 là phiên bản mới nhất tính đến thời điểm đó, với các thay đổi đáng chú ý trong ngôn ngữ và thư viện chuẩn.
* Python đã trải qua một sự phát triển rất lớn từ khi được tạo ra, từ một ngôn ngữ đơn giản cho giáo dục đến một công cụ mạnh mẽ và linh hoạt được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau như phát triển web, khoa học dữ liệu, trí tuệ nhân tạo, và nhiều ứng dụng khác.

1.2 Đặc trưng của ngôn ngữ lập trình Python

1. Cú Pháp Đơn Giản và Dễ Hiểu:  
 Python được thiết kế để dễ đọc và dễ hiểu, với cú pháp gần giống với ngôn ngữ tự nhiên. Điều này làm cho Python trở thành lựa chọn lý tưởng cho cả người mới bắt đầu và những người có kinh nghiệm trong lập trình.  
2.Linh Hoạt và Đa Dạng :  
 Python là một ngôn ngữ lập trình đa mục đích, có thể được sử dụng cho nhiều mục đích khác nhau từ phát triển web, khoa học dữ liệu, trí tuệ nhân tạo đến viết script và automation. Sự linh hoạt này giúp Python trở thành một công cụ hữu ích cho nhiều dự án và ứng dụng.  
3. Hệ Sinh Thái Thư Viện Phong Phú:  
 Python đi kèm với một hệ sinh thái thư viện phong phú, cung cấp cho người lập trình các công cụ và thư viện để giải quyết hầu hết các vấn đề trong lập trình. Các thư viện như NumPy, Pandas, Matplotlib, và TensorFlow hỗ trợ cho các lĩnh vực như xử lý dữ liệu, tính toán khoa học, đồ họa và trí tuệ nhân tạo.  
4. Cộng Đồng Lập Trình Viên Lớn:  
 Python có một cộng đồng lập trình viên rộng lớn, tích cực đóng góp vào việc phát triển ngôn ngữ và cung cấp hỗ trợ, tài liệu và các nguồn tài nguyên cho người mới học và người lập trình viên có kinh nghiệm.  
5. Hỗ Trợ Đa Nền Tảng:  
 Python có thể chạy trên nhiều hệ điều hành khác nhau như Windows, macOS và Linux, giúp cho việc phát triển và triển khai ứng dụng trở nên dễ dàng và linh hoạt hơn.  
Kết Luận:  
 Python là một ngôn ngữ lập trình mạnh mẽ, linh hoạt và dễ sử dụng, phù hợp cho cả người mới học lập trình và những lập trình viên có kinh nghiệm. Với đặc điểm đa mục đích, thư viện phong phú và cộng đồng lập trình viên lớn, Python tiếp tục là một trong những ngôn ngữ lập trình được ưa chuộng nhất hiện nay.

## 2. Tìm hiểu về ngôn ngữ lập trình Python

2.1 Cú pháp

[Python](https://topdev.vn/blog/python-la-gi/) không hỗ trợ các ký tự đặc biệt chẳng hạn như @, $ và % bên trong các định danh. Python là một ngôn ngữ lập trình phân biệt chữ hoa- chữ thường, do đó định danh UCODE và ucode là hai định danh hoàn toàn khác nhau trong lập trình Python.

Python không cung cấp các dấu ngoặc ôm ({}) để chỉ các khối code cho định nghĩa lớp hoặc hàm hoặc điều khiển luồng. Các khối code được nhận biết bởi độ thụt dòng code (indentation) trong lập trình Python và đây là điều bắt buộc.

Số khoảng trống trong độ thụt dòng là biến đổi, nhưng tất cả các lệnh bên trong khối phải được thụt cùng một số lượng khoảng trống như nhau

Các lệnh trong Python có một nét đặc trưng là kết thúc với một newline (dòng mới). Tuy nhiên, Python cho phép sử dụng ký tự để chỉ rõ sự liên tục dòng.

Python chấp nhận trích dẫn đơn (‘), kép (“) và trích dẫn tam (”’ hoặc “””) để biểu thị các hằng chuỗi, miễn là các trích dẫn này có cùng kiểu mở và đóng.

Python hỗ trợ hai kiểu comment đó là comment đơn dòng và đa dòng. Trong lập trình Python, một dấu #, mà không ở bên trong một hằng chuỗi nào, bắt đầu một comment đơn dòng. Tất cả ký tự ở sau dấu # và kéo dài cho đến hết dòng đó thì được coi là một comment và được bỏ qua bởi trình thông dịch.

2.2 Các kiểu dữ liệu trong python

Trong Python, có nhiều kiểu dữ liệu khác nhau giúp bạn lưu trữ và thao tác với các loại dữ liệu khác nhau. Dưới đây là các kiểu dữ liệu cơ bản và phổ biến trong Python:

1. Kiểu Dữ Liệu Số (Numeric Types)

- int: Số nguyên, ví dụ: `1`, `100`, `-10`

- float: Số thực, ví dụ: `1.5`, `0.001`, `-3.14`

- complex: Số phức, ví dụ: `3+5j`, `2-2j`

2. Kiểu Dữ Liệu Chuỗi (Text Type)

- str: Chuỗi ký tự, ví dụ: `"hello"`, `'world'`, `"123"`

3. Kiểu Dữ Liệu Chuỗi Byte (Binary Types)

- bytes: Chuỗi byte không thay đổi, ví dụ: `b'hello'`

- bytearray: Chuỗi byte có thể thay đổi, ví dụ: `bytearray(b'hello')`

- memoryview: Khung nhìn bộ nhớ, ví dụ: `memoryview(bytes(5))`

4. Kiểu Dữ Liệu Dãy (Sequence Types)

- list: Danh sách các phần tử có thể thay đổi, ví dụ: `[1, 2, 3]`, `['a', 'b', 'c']`

- tuple: Bộ các phần tử không thay đổi, ví dụ: `(1, 2, 3)`, `('a', 'b', 'c')`

- range: Chuỗi số, thường được sử dụng trong vòng lặp, ví dụ: `range(5)`, `range(1, 10, 2)`

5. Kiểu Dữ Liệu Bộ Sưu Tập (Set Types)

- set: Tập hợp các phần tử duy nhất không có thứ tự, ví dụ: `{1, 2, 3}`, `{'a', 'b', 'c'}`

- frozenset: Tập hợp các phần tử duy nhất không có thứ tự và không thể thay đổi, ví dụ: `frozenset([1, 2, 3])`

6. Kiểu Dữ Liệu Ánh Xạ (Mapping Type)

- dict: Từ điển chứa các cặp khóa-giá trị, ví dụ: `{'name': 'Alice', 'age': 25}`, `{1: 'one', 2: 'two'}`

7. Kiểu Dữ Liệu Boolean (Boolean Type)

- bool: Giá trị chân lý, chỉ có hai giá trị `True` và `False`

8. Kiểu Dữ Liệu Không Có (None Type)

- NoneType: Đối tượng không có giá trị, chỉ có một giá trị là `None`

Các Ví Dụ Cụ Thể

```python

# Kiểu int

x = 5

print(type(x)) # Output: <class 'int'>

# Kiểu float

y = 3.14

print(type(y)) # Output: <class 'float'>

# Kiểu complex

z = 1 + 2j

print(type(z)) # Output: <class 'complex'>

# Kiểu str

s = "hello"

print(type(s)) # Output: <class 'str'>

# Kiểu list

lst = [1, 2, 3]

print(type(lst)) # Output: <class 'list'>

# Kiểu tuple

t = (1, 2, 3)

print(type(t)) # Output: <class 'tuple'>

# Kiểu set

st = {1, 2, 3}

print(type(st)) # Output: <class 'set'>

# Kiểu frozenset

fst = frozenset([1, 2, 3])

print(type(fst)) # Output: <class 'frozenset'>

# Kiểu dict

d = {'key': 'value'}

print(type(d)) # Output: <class 'dict'>

# Kiểu bool

b = True

print(type(b)) # Output: <class 'bool'>

# Kiểu NoneType

n = None

print(type(n)) # Output: <class 'NoneType'>

```

Python còn hỗ trợ nhiều kiểu dữ liệu phức tạp và cấu trúc dữ liệu khác, nhưng các kiểu trên là cơ bản và thường dùng nhất.

2.3 Cấu trúc dữ liệu trong python

Trong Python, có nhiều loại cấu trúc dữ liệu khác nhau giúp bạn tổ chức, lưu trữ và thao tác với dữ liệu một cách hiệu quả. Dưới đây là các loại cấu trúc dữ liệu chính:

1. List (Danh sách)

- \*\*Đặc điểm\*\*: Là một dãy các phần tử có thể thay đổi, có thể chứa các kiểu dữ liệu khác nhau.

- \*\*Cú pháp\*\*: `lst = [1, 2, 3]`

- \*\*Ví dụ\*\*:

```python

lst = [1, 2, 3, "hello", [4, 5]]

lst.append(6)

lst[1] = 9

print(lst) # Output: [1, 9, 3, 'hello', [4, 5], 6]

```

2. Tuple (Bộ)

- \*\*Đặc điểm\*\*: Là một dãy các phần tử không thể thay đổi, có thể chứa các kiểu dữ liệu khác nhau.

- \*\*Cú pháp\*\*: `tup = (1, 2, 3)`

- \*\*Ví dụ\*\*:

```python

tup = (1, 2, 3, "hello", [4, 5])

print(tup[1]) # Output: 2

```

3. Set (Tập hợp)

- \*\*Đặc điểm\*\*: Là một tập hợp các phần tử duy nhất không có thứ tự, không chứa các phần tử trùng lặp.

- \*\*Cú pháp\*\*: `st = {1, 2, 3}`

- \*\*Ví dụ\*\*:

```python

st = {1, 2, 3, 4, 5}

st.add(6)

st.remove(3)

print(st) # Output: {1, 2, 4, 5, 6}

```

4. Frozenset (Tập hợp không thay đổi)

- \*\*Đặc điểm\*\*: Là một phiên bản không thể thay đổi của set.

- \*\*Cú pháp\*\*: `fst = frozenset([1, 2, 3])`

- \*\*Ví dụ\*\*:

```python

fst = frozenset([1, 2, 3, 4, 5])

print(fst) # Output: frozenset({1, 2, 3, 4, 5})

```

5. Dict (Từ điển)

- \*\*Đặc điểm\*\*: Là một tập hợp các cặp khóa-giá trị.

- \*\*Cú pháp\*\*: `d = {'key1': 'value1', 'key2': 'value2'}`

- \*\*Ví dụ\*\*:

```python

d = {'name': 'Alice', 'age': 25}

d['age'] = 26

d['address'] = '123 Main St'

print(d) # Output: {'name': 'Alice', 'age': 26, 'address': '123 Main St'}

```

6. Các cấu trúc dữ liệu từ thư viện `collections`

Python cung cấp một số cấu trúc dữ liệu bổ sung trong module `collections`, bao gồm:

6.1. deque (Double-ended Queue)

- \*\*Đặc điểm\*\*: Là một deque cung cấp các phương thức nhanh chóng cho cả hai đầu của deque.

- \*\*Cú pháp\*\*: `from collections import deque`

- \*\*Ví dụ\*\*:

```python

from collections import deque

dq = deque([1, 2, 3])

dq.appendleft(0)

dq.append(4)

print(dq) # Output: deque([0, 1, 2, 3, 4])

```

6.2. namedtuple

- \*\*Đặc điểm\*\*: Là một tuple nhưng có tên các trường.

- \*\*Cú pháp\*\*: `from collections import namedtuple`

- \*\*Ví dụ\*\*:

```python

from collections import namedtuple

Point = namedtuple('Point', ['x', 'y'])

p = Point(1, 2)

print(p.x, p.y) # Output: 1 2

```

6.3. Counter

- \*\*Đặc điểm\*\*: Là một dict chuyên đếm số lượng xuất hiện của các phần tử.

- \*\*Cú pháp\*\*: `from collections import Counter`

- \*\*Ví dụ\*\*:

```python

from collections import Counter

cnt = Counter(['a', 'b', 'c', 'a', 'b', 'b'])

print(cnt) # Output: Counter({'b': 3, 'a': 2, 'c': 1})

```

6.4. defaultdict

- \*\*Đặc điểm\*\*: Là một dict cho phép chỉ định một giá trị mặc định cho các khóa chưa có giá trị.

- \*\*Cú pháp\*\*: `from collections import defaultdict`

- \*\*Ví dụ\*\*:

```python

from collections import defaultdict

dd = defaultdict(int)

dd['a'] += 1

print(dd) # Output: defaultdict(<class 'int'>, {'a': 1})

```

7. Các cấu trúc dữ liệu từ module `array`

Python cũng cung cấp một module `array` để làm việc với mảng số nguyên và số thực hiệu quả hơn về mặt bộ nhớ.

- \*\*Cú pháp\*\*: `from array import array`

- \*\*Ví dụ\*\*:

```python

from array import array

arr = array('i', [1, 2, 3, 4])

arr.append(5)

print(arr) # Output: array('i', [1, 2, 3, 4, 5])

```

Các cấu trúc dữ liệu này cung cấp nhiều công cụ mạnh mẽ để làm việc với dữ liệu trong Python, giúp bạn lựa chọn giải pháp phù hợp nhất cho từng tình huống cụ thể.

2.4 Vòng lặp và câu điều kiện

Trong Python, vòng lặp và các câu điều kiện là những cấu trúc điều khiển cơ bản cho phép bạn thực hiện các thao tác lặp lại và kiểm tra các điều kiện logic. Dưới đây là các vòng lặp và câu điều kiện phổ biến trong Python.

## Vòng Lặp (Loops)

1. Vòng lặp `for`

Vòng lặp `for` được sử dụng để lặp qua một dãy (như danh sách, tuple, từ điển, tập hợp, hoặc chuỗi).

Cú pháp:

```python

for element in iterable:

# Thực hiện một số thao tác với element

```

Ví dụ:

```python

# Lặp qua danh sách

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

# Lặp qua chuỗi

for char in "hello":

print(char)

# Lặp qua dãy số sử dụng range()

for i in range(5):

print(i) # Output: 0, 1, 2, 3, 4

```

2. Vòng lặp `while`

Vòng lặp `while` tiếp tục lặp cho đến khi điều kiện logic được kiểm tra trở thành False.

Cú pháp:

```python

while condition:

# Thực hiện một số thao tác

```

Ví dụ:

```python

# Lặp cho đến khi biến đếm đạt đến 5

count = 0

while count < 5:

print(count)

count += 1 # Output: 0, 1, 2, 3, 4

```

3. Câu lệnh điều khiển vòng lặp: `break`, `continue`, `else`

- \*\*`break`\*\*: Dừng vòng lặp ngay lập tức.

- \*\*`continue`\*\*: Bỏ qua các câu lệnh còn lại trong vòng lặp hiện tại và tiếp tục với vòng lặp kế tiếp.

- \*\*`else`\*\*: Được thực hiện khi vòng lặp hoàn tất một cách bình thường, không bị dừng bởi `break`.

Ví dụ:

```python

# Sử dụng break

for i in range(5):

if i == 3:

break

print(i) # Output: 0, 1, 2

# Sử dụng continue

for i in range(5):

if i == 3:

continue

print(i) # Output: 0, 1, 2, 4

# Sử dụng else với for

for i in range(5):

print(i)

else:

print("Loop finished without break.") # Output: 0, 1, 2, 3, 4 và "Loop finished without break."

```

## Câu Điều Kiện (Conditional Statements)

1. Câu điều kiện `if`

Câu điều kiện `if` được sử dụng để kiểm tra một điều kiện logic và thực thi các câu lệnh tương ứng nếu điều kiện đó là True.

Cú pháp:

```python

if condition:

# Thực hiện một số thao tác nếu condition là True

```

Ví dụ:

```python

x = 10

if x > 5:

print("x is greater than 5") # Output: x is greater than 5

```

2. Câu điều kiện `if-else`

Câu điều kiện `if-else` mở rộng câu điều kiện `if` bằng cách thêm một khối lệnh để thực thi nếu điều kiện là False.

Cú pháp:

```python

if condition:

# Thực hiện một số thao tác nếu condition là True

else:

# Thực hiện một số thao tác nếu condition là False

```

Ví dụ:

```python

x = 10

if x > 15:

print("x is greater than 15")

else:

print("x is not greater than 15") # Output: x is not greater than 15

```

3. Câu điều kiện `if-elif-else`

Câu điều kiện `if-elif-else` được sử dụng để kiểm tra nhiều điều kiện khác nhau. `elif` là viết tắt của "else if".

Cú pháp:

```python

if condition1:

# Thực hiện một số thao tác nếu condition1 là True

elif condition2:

# Thực hiện một số thao tác nếu condition2 là True

else:

# Thực hiện một số thao tác nếu không có điều kiện nào ở trên là True

```

Ví dụ:

```python

x = 10

if x > 15:

print("x is greater than 15")

elif x > 5:

print("x is greater than 5 but not greater than 15") # Output: x is greater than 5 but not greater than 15

else:

print("x is 5 or less")

```

4. Các biểu thức điều kiện lồng nhau

Bạn có thể lồng các câu điều kiện bên trong nhau để kiểm tra các điều kiện phức tạp hơn.

Ví dụ:

```python

x = 10

y = 20

if x > 5:

if y > 15:

print("x is greater than 5 and y is greater than 15") # Output: x is greater than 5 and y is greater than 15

```

Ví dụ tổng hợp

Kết hợp vòng lặp và các câu điều kiện để tạo ra các logic phức tạp hơn:

```python

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for number in numbers:

if number % 2 == 0:

print(f"{number} is even")

else:

print(f"{number} is odd")

if number > 5:

break # Dừng vòng lặp khi gặp số lớn hơn 5

```

Kết quả sẽ hiển thị:

```

1 is odd

2 is even

3 is odd

4 is even

5 is odd

6 is even

```

Với các cấu trúc điều khiển này, bạn có thể tạo ra các chương trình Python linh hoạt và mạnh mẽ để giải quyết nhiều bài toán khác nhau.

3. Một số thư viện dành cho thực hiện xử lý, mô hình hóa dữ liệu.

4. Thuật toán và phương pháp Machine Learning kinh điển.