

LẬP TRÌNH PYTHON CƠ BẢN



Tên học phần	Lập trình Python cơ bản
Thời lượng	Lý thuyết: 30 tiết; Thực hành: 30 giờ.
Số bài kiểm tra	02 bài
Điểm chuyên cần	Điểm danh
Hình thức kiểm tra	Trên máy, 45 phút
Hình thức thi	Trên máy, 60 phút
Tài liệu:	[1]. Slide bài giảng. [2]. Bài thực hành. [3]. Các tài liệu trên mạng Internet.

#	TÊN BÀI	SỐ TIẾT
1	Giới thiệu về Python và các lệnh cơ bản	3
2	Hàm và Gói	3
3	Các cấu trúc dữ liệu	3
4	Các cấu trúc dữ liệu (tiếp)	3
5	Xử lý file, nhập xuất dữ liệu	3
6	Xử lý file, nhập xuất dữ liệu (tiếp)	3
7	Lập trình hướng đối tượng với Python	3
8	Lập trình hướng đối tượng với Python (tiếp)	3
9	Lập trình hướng đối tượng với Python (tiếp)	3
10	Lập trình hướng đối tượng với Python (tiếp)	3



#	TÊN BÀI	SỐ TIẾT
1	Cài đặt và sử dụng Python cơ bản	3
2	Sử dụng và phân biệt Hàm và Gói	3
3	Thao tác trên các cấu trúc dữ liệu	3
4	Ôn tập bài 6,7,8	3
5	Ôn tập bài 6,7,8 (tiếp)	3
6	Xử lý file dữ liệu và nhập xuất dữ liệu	3
7	Lập trình hướng đối tượng với Python	3
8	Lập trình hướng đối tượng với Python (tiếp)	3
9	Ôn tập bài 10,11	3
10	Ôn tập bài 10,11 (tiếp)	3



BÀI 1

PYTHON VÀ CÁC LỆNH CƠ BẢN



- Ngôn ngữ đơn giản, dễ tiếp cận
- Ngôn ngữ phổ biến hàng đầu
- Làm được nhiều việc: Machine Learning, Web development, Data analysis, Scientific research, Automation,...
- Sử dụng trong nhiều công ty nổi tiếng: Google, Meta, Netflix,...



- **Ưu điểm:**
 - Python là một ngôn ngữ lập trình dễ đọc, dễ sử dụng
 - Khả năng mở rộng và nhúng vào các ngôn ngữ khác
 - Tiết kiệm thời gian
 - Sửa đổi và cải tiến mã nguồn
 - Có thể chạy các chương trình Python trên nhiều nền tảng khác nhau.
 - Python cho phép bạn tập trung vào giải quyết vấn đề, chứ không phải lo lắng về cú pháp phức tạp hay cứng nhắc.



- **Nhược điểm:**

- Chạy chậm hơn các ngôn ngữ biên dịch: Python là một ngôn ngữ thông dịch, tức là nó phải dịch mã nguồn sang mã máy khi chạy. Điều này làm cho Python chạy chậm hơn các ngôn ngữ biên dịch như C hoặc C++.
- Dùng nhiều bộ nhớ hơn: Python dùng một hệ thống quản lý bộ nhớ tự động, gọi là bộ thu gom rác (garbage collector), để giải phóng những vùng nhớ không cần thiết. Tuy nhiên, điều này cũng khiến Python dùng nhiều bộ nhớ hơn và khó kiểm soát hơn.
- Giới hạn của người lập trình: Python có một số quy định và quy tắc về cú pháp và định dạng mã, như phải thụt lề, không có dấu ngoặc nhọn,... Điều này có thể giảm sự sáng tạo và tự do của người lập trình.



1. Từ khóa và định danh
2. Chú thích
3. Biến, hằng
4. Kiểu dữ liệu và chuyển đổi kiểu dữ liệu
5. Nhập/Xuất
6. Toán tử
7. Cấu trúc điều khiển



Từ khóa Python

- Từ khóa là những từ được xác định, có ý nghĩa riêng đối với trình dịch.
- Từ khóa được sử dụng để xác định cú pháp và cấu trúc của Python. Không sử dụng từ khóa để đặt tên cho biến, hàm hoặc các định danh khác.
- Tất cả các từ khóa ngoại trừ True, False và None đều viết thường.



Từ khóa Python

Python Keywords List				
False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Định danh trong Python

- Giá trị định danh không được trùng với từ khóa.
- Giá trị định danh phân biệt chữ hoa chữ thường.
- Định danh phải bắt đầu bằng một chữ cái hoặc _. Chữ cái đầu tiên của mã định danh không được là chữ số.
- Đó là một quy ước để bắt đầu một định danh với một chữ cái thay vì _.
- Định danh không được phép chứa khoảng trắng.
- Không được sử dụng các ký hiệu đặc biệt như **!**, **@**, **#**, **\$**, v.v.



- Chú thích trong Python
 - Sử dụng ký tự “#”

```
# khai báo một biến  
name = 'Anh'  
  
# hiển thị giá trị của biến  
print(name) # Anh
```

- Phím tắt: **Ctrl + /** (Windows) hoặc **Cmd + /** (Mac)



- Chú thích trong Python

- Có thể sử dụng `""" <comment> """` để chú thích trên nhiều dòng
- 3 dấu `'` hoặc `"""` viết liền nhau

```
""" Chú thích  
trên nhiều dòng"""
```

```
name = input('Enter your name: ')  
print(name)
```



Biến trong Python

- Trong lập trình, biến đại diện cho một chỗ chứa (vùng nhớ) để lưu trữ dữ liệu.
- Gán giá trị cho biến:

```
number = 10
```

- Thay đổi giá trị của biến:

```
number = 20
```

- Gán giá trị cho nhiều biến:

```
a, b, c = 5, 3.2, 'Hello'
```



Hằng trong Python

- Trong Python, hằng số thường được khai báo và gán trong một mô-đun (một tệp mới chứa các biến, hàm, v.v.).
- Tạo tệp **constant.py**:

```
# định nghĩa hằng số  
PI = 3.14  
GRAVITY = 9.8
```

- Tạo tệp **main.py**:

```
# nhập tệp constant.py  
import constant  
  
print(constant.PI) # 3.14  
  
print(constant.GRAVITY) # 9.8
```



Kiểu dữ liệu trong Python

- Kiểu dữ liệu trong Python chính là các lớp và các biến là các đối tượng thuộc các lớp tương ứng với kiểu dữ liệu đó.

Kiểu dữ liệu	Lớp	Mô tả
Số	int, float, complex	xử lý các giá trị số
Chuỗi	str	xử lý chuỗi ký tự
Tuần tự	list, tuple, range	xử lý danh sách các đối tượng
Ánh xạ	dict	xử lý dữ liệu dạng key, value
Logic	bool	xử lý dữ liệu True và False
Tập hợp	set, frozenset	xử lý dữ liệu về tập hợp



Dữ liệu số (numeric) trong Python

- int – số nguyên có dấu với độ dài không giới hạn.
- float – số thực có chính xác lên đến 15 chữ số thập phân.
- complex – số phức.

```
num1 = 5
print(num1, 'is of type', type(num1))

num2 = 2.0
print(num2, 'is of type', type(num2))

num3 = 1+2j
print(num3, 'is of type', type(num3))
```

```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is of type <class 'complex'>
```



Danh sách (list) trong Python

- Danh sách là một tập hợp có thứ tự gồm các phần tử thuộc nhiều kiểu dữ liệu khác nhau.

```
languages = ["Swift", "Java", "Python"]
```

```
# truy cập vào phần tử có chỉ số 0
```

```
print(languages[0]) # Swift
```

```
# truy cập vào phần tử có chỉ số 2
```

```
print(languages[2]) # Python
```


Tuple trong Python

- Tuple là một tập hợp các phần tử được sắp xếp giống như một danh sách. Sự khác biệt duy nhất là các bộ dữ liệu không thể thay đổi được. Các bộ dữ liệu một khi đã được tạo thì không thể sửa đổi được

```
# tạo một tuple  
product = ('Microsoft', 'Xbox', 499.99)  
  
# truy cập vào phần tử có chỉ số 0  
print(product[0]) # Microsoft  
  
# truy cập vào phần tử có chỉ số 1  
print(product[1]) # Xbox
```


Chuỗi (string) trong Python

- Chuỗi là một tập hợp các ký tự được biểu diễn bên trong cặp dấu ngoặc đơn hoặc dấu ngoặc kép.

```
name = 'Python'  
print(name) # Python
```

```
message = "Python for beginners"  
print(message) # Python for beginners
```


Tập hợp (set) trong Python

- Tập hợp là một bộ dữ liệu không được sắp xếp, các phần tử trong tập hợp là duy nhất.

```
# tạo một tập hợp có tên student_id  
student_id = {112, 114, 116, 118, 115}  
  
# hiển thị giá trị của student_id  
print(student_id) # {112, 114, 115, 116, 118}  
  
# hiển thị kiểu của student_id  
print(type(student_id)) # <class 'set'>
```


Từ điển (dictionary) trong Python

- Từ điển là một tập hợp các phần tử được sắp xếp theo thứ tự. Các phần tử được lưu trữ theo cặp **khóa/giá trị** (**key/value**).
- Có thể truy cập các **giá trị** trong từ điển thông qua **khóa**.

```
# tạo từ điển có tên capital_city  
capital_city = {'Nepal': 'Kathmandu', 'Italy': 'Rome', 'England': 'London'}  
  
print(capital_city['Nepal']) # Kathmandu  
  
print(capital_city['Kathmandu']) # xảy ra lỗi
```



Chuyển đổi kiểu dữ liệu

- Là quá trình chuyển đổi dữ liệu từ kiểu này sang kiểu khác.
- Có 2 loại chuyển đổi kiểu dữ liệu trong Python:
 - Chuyển đổi ngầm (Implicit Conversion) – chuyển đổi tự động
 - Chuyển đổi rõ ràng (Explicit Conversion) – chuyển đổi thủ công



Chuyển đổi kiểu dữ liệu

- Chuyển đổi ngầm (Implicit Conversion)

```
integer_number = 123
```

```
float_number = 1.23
```

```
new_number = integer_number + float_number
```

```
# hiển thị giá trị mới và kiểu dữ liệu của kết quả
```

```
print("Value:",new_number) # Value: 124.23
```

```
print("Data Type:",type(new_number)) # Data Type: <class 'float'>
```


Chuyển đổi kiểu dữ liệu

- Chuyển đổi rõ ràng (Explicit Conversion)

```
num_string = '12'  
num_integer = 23  
  
print("Kiểu dữ liệu của num_string:", type(num_string))  
  
# chuyển đổi rõ ràng  
num_string = int(num_string)  
  
print("Kiểu dữ liệu của num_string :", type(num_string))  
  
num_sum = num_integer + num_string  
  
print("Sum:", num_sum)  
print("Kiểu dữ liệu của num_sum:", type(num_sum))
```

```
Kiểu dữ liệu của num_string: <class 'str'>  
Kiểu dữ liệu của num_string: <class 'int'>  
Sum: 35  
Kiểu dữ liệu của num_sum: <class 'int'>
```



Chuyển đổi kiểu dữ liệu

- Lưu ý:
 - Chuyển kiểu ngầm định được trình thông dịch Python thực hiện tự động
 - Chuyển kiểu ngầm định luôn chuyển kiểu dữ liệu nhỏ hơn thành kiểu dữ liệu lớn hơn để tránh được việc mất mát dữ liệu.
 - Chuyển kiểu rõ ràng (còn gọi là Type Casting), kiểu dữ liệu được chuyển đổi bởi lập trình viên.
 - Chuyển kiểu rõ ràng có khả năng gây ra mất mát dữ liệu.



Python Output

- Sử dụng **print()**
- Cú pháp:
 - **object** – giá trị hiển thị.
 - **sep** (tùy chọn) – chuỗi ngăn cách các đối tượng bên trong print().
 - **end** (tùy chọn) – thêm giá trị cụ thể khi kết thúc lệnh print().
 - **file** (tùy chọn) – nơi giá trị được in ra. Mặc định là sys.stdout (màn hình)
 - **flush** (tùy chọn) – xóa bộ nhớ đệm. Mặc định: False.

```
print(  
    *objects,  
    sep=' ',  
    end='\n',  
    file=sys.stdout,  
    flush=False  
)
```



Python Input

- Sử dụng **input()**
- Cú pháp: `input(prompt)`

```
# sử dụng input() lấy giá trị từ người dùng  
num = input('Nhập một số: ')  
  
print('Số đã nhập:', num)  
print('Kiểu dữ liệu nhập:', type(num))
```

Nhập một số: 10

Số đã nhập: 10

Kiểu dữ liệu nhập: <class 'str'>

- *Lưu ý: Giá trị nhập vào có kiểu dữ liệu là chuỗi.*

```
num = int(input('Nhập một số: '))
```



- **Các loại toán tử**
 - Toán tử số học
 - Toán tử gán
 - Toán tử so sánh
 - Toán tử logic
 - Toán tử trên bit
 - Toán tử đặc biệt



- Toán tử số học

Toán tử	Chức năng	Ví dụ
+	Cộng	$5 + 2 = 7$
-	Trừ	$4 - 2 = 2$
*	Nhân	$2 * 3 = 6$
/	Chia	$4 / 2 = 2$
//	Chia lấy nguyên	$10 // 3 = 3$
%	Chia lấy dư	$5 \% 2 = 1$
**	Lũy thừa	$4 ** 2 = 16$



- Toán tử gán

Toán tử	Ý nghĩa	Ví dụ
=	Toán tử gán	$a = 7$
+=	Gán tổng	$a += 1 \# a = a + 1$
-=	Gán hiệu	$a -= 3 \# a = a - 3$
*=	Gán tích	$a *= 4 \# a = a * 4$
/=	Gán thương	$a /= 3 \# a = a / 3$
%=	Gán lấy dư	$a \% = 10 \# a = a \% 10$
**=	Gán lũy thừa	$a ** = 10 \# a = a ** 10$



- Toán tử so sánh

Toán tử	Ý nghĩa	Ví dụ
==	bằng	3 == 5 trả về False
!=	không bằng	3 != 5 trả về True
>	lớn hơn	3 > 5 trả về False
<	nhỏ hơn	3 < 5 trả về True
>=	lớn hơn hoặc bằng	3 >= 5 trả về False
<=	nhỏ hơn hoặc bằng	3 <= 5 trả về True



- Toán tử logic

Toán tử	Ví dụ	Ý nghĩa
and	a and b	Logic AND: Chỉ đúng nếu cả 2 toán hạng đều đúng
or	a or b	Logic OR: Đúng nếu ít nhất một trong các toán hạng là Đúng
not	not a	Logic NOT: Đúng nếu toán hạng là Sai và ngược lại.



- **Toán tử trên bit**

- Với $x = 10$ (0000 1010) và $y = 4$ (0000 0100)

Toán tử	Ý nghĩa	Ví dụ
&	Bitwise AND	$x \& y = 0$ (0000 0000)
	Bitwise OR	$x y = 14$ (0000 1110)
~	Bitwise NOT	$\sim x = -11$ (1111 0101)
^	Bitwise XOR	$x \wedge y = 14$ (0000 1110)
>>	Bitwise right shift	$x >> 2 = 2$ (0000 0010)
<<	Bitwise left shift	$x << 2 = 40$ (0010 1000)



- Toán tử đặc biệt

Toán tử	Ý nghĩa	Ví dụ
is	Đúng nếu các toán hạng giống hệt nhau (tham chiếu đến cùng một đối tượng)	x is True
is not	Đúng nếu các toán hạng không giống nhau (không tham chiếu đến cùng một đối tượng)	x is not True



1. Nhập vào từ bàn phím hai số nguyên a, b. Tính và in ra màn hình tổng, hiệu, tích, thương của a và b:

- Mỗi kết quả in trên 1 dòng.
- Kết quả của thương: là số thực có độ chính xác 3 chữ số hàng thập phân.

2. Nhập vào tọa độ của hai điểm A(x1, y1) và B(x2, y2). Tính và in ra khoảng cách Euclidean giữa A và B:

$$d(A, B) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$



- Cấu trúc rẽ nhánh
 - Câu lệnh **if**

Condition is True

```
number = 10  
if number > 0:  
    # code  
  
# code after if
```

Condition is False

```
number = -5  
if number > 0:  
    # code  
  
# code after if
```



- Cấu trúc rẽ nhánh
 - Câu lệnh **if...else**

Condition is True

```
number = 10
if number > 0:
    # code

else:
    # code

# code after if
```

Condition is False

```
number = -5
if number > 0:
    # code

else:
    # code

# code after if
```



- Cấu trúc rẽ nhánh
 - Câu lệnh **if...elif...else**

1st Condition is True

```
let number = 5
if number > 0 :
    # code
elif number < 0 :
    # code
else :
    # code
# code after if
```

2nd Condition is True

```
let number = -5
if number > 0 :
    # code
elif number < 0 :
    # code
else :
    # code
# code after if
```

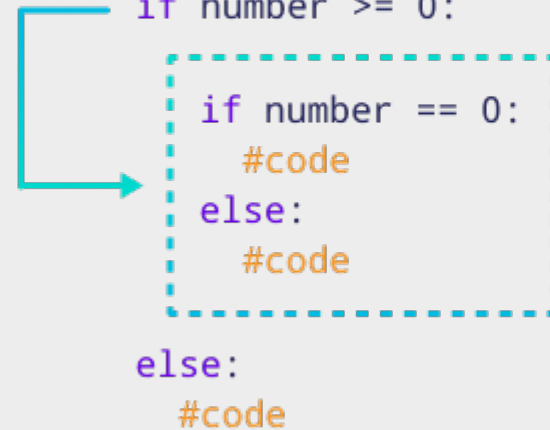
All Conditions are False

```
let number = 0
if number > 0 :
    # code
elif number < 0 :
    # code
else :
    # code
# code after if
```

- Cấu trúc rẽ nhánh
 - Câu lệnh **if** lồng nhau

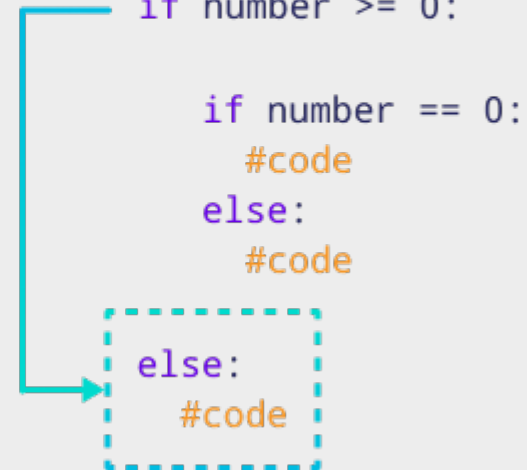
Outer if Condition is True

```
number = 5
if number >= 0:
    if number == 0:
        #code
    else:
        #code
else:
    #code
```



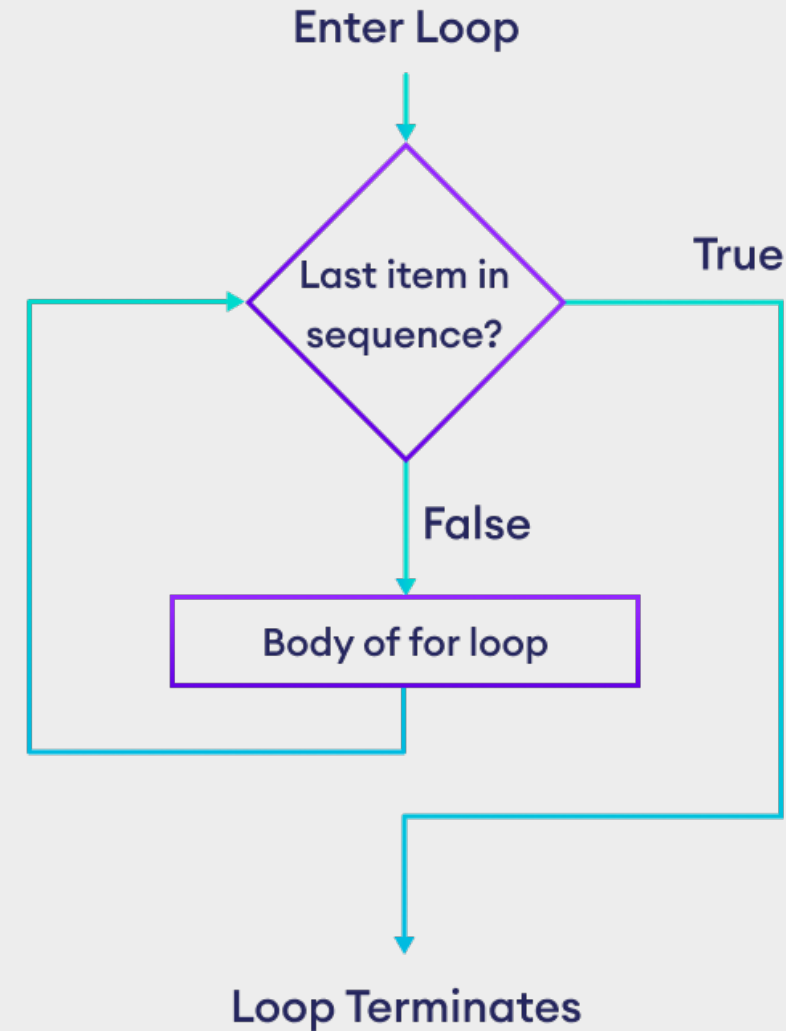
Outer if Condition is False

```
number = -5
if number >= 0:
    if number == 0:
        #code
    else:
        #code
else:
    #code
```



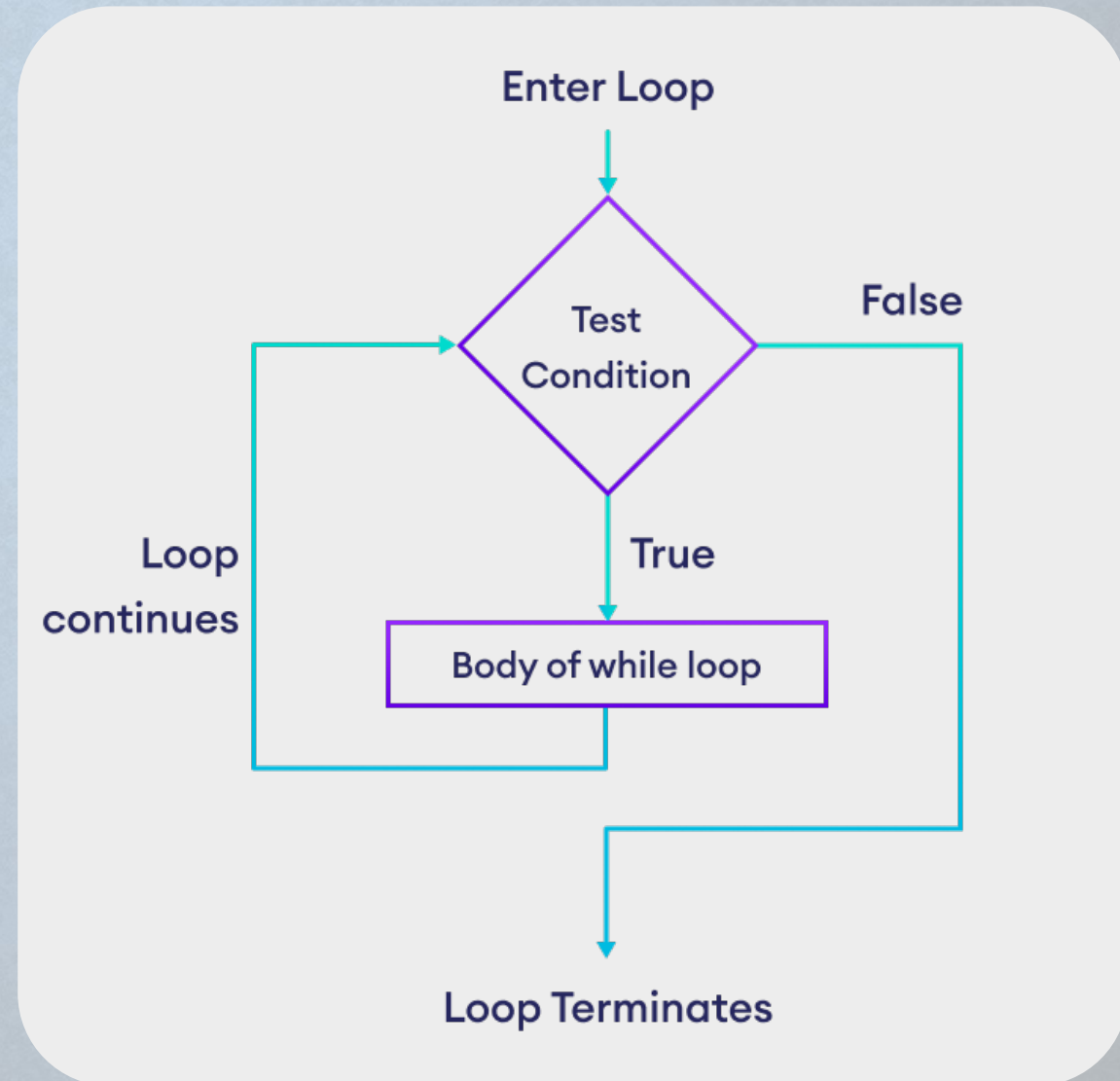
- Cấu trúc lặp
 - Câu lệnh lặp **for**

```
for val in sequence:  
    # statement(s)
```



- Cấu trúc lặp
 - Câu lệnh lặp **while**

while condition:
body of while loop



- Cấu trúc lặp
- break và continue

```
for val in sequence:  
    # code  
    if condition:  
        break
```



code

```
while condition:  
    # code  
    if condition:  
        break
```



code

```
for val in sequence:  
    # code  
    if condition:  
        continue
```



code

```
while condition:  
    # code  
    if condition:  
        continue
```



code



1. Giải phương trình bậc 2.
2. Nhập vào một số nguyên dương n. kiểm tra xem n có phải là số nguyên tố hay không?
3. Nhập vào một số nguyên n cho tới khi n thuộc [20, 30], nhập vào số thực x. Tính và in ra:

$$P = 2022 x^n + \frac{1}{x} + \frac{2}{x^2} + \dots + \frac{n}{x^n}$$



KẾT THÚC

