

BUỔI I. Giới thiệu Python và Môi trường lập trình

Python for AI

ThS. Phạm Đức Cường

cuongpd@ptit.edu.vn



Posts and Telecommunications
Institute of Technology

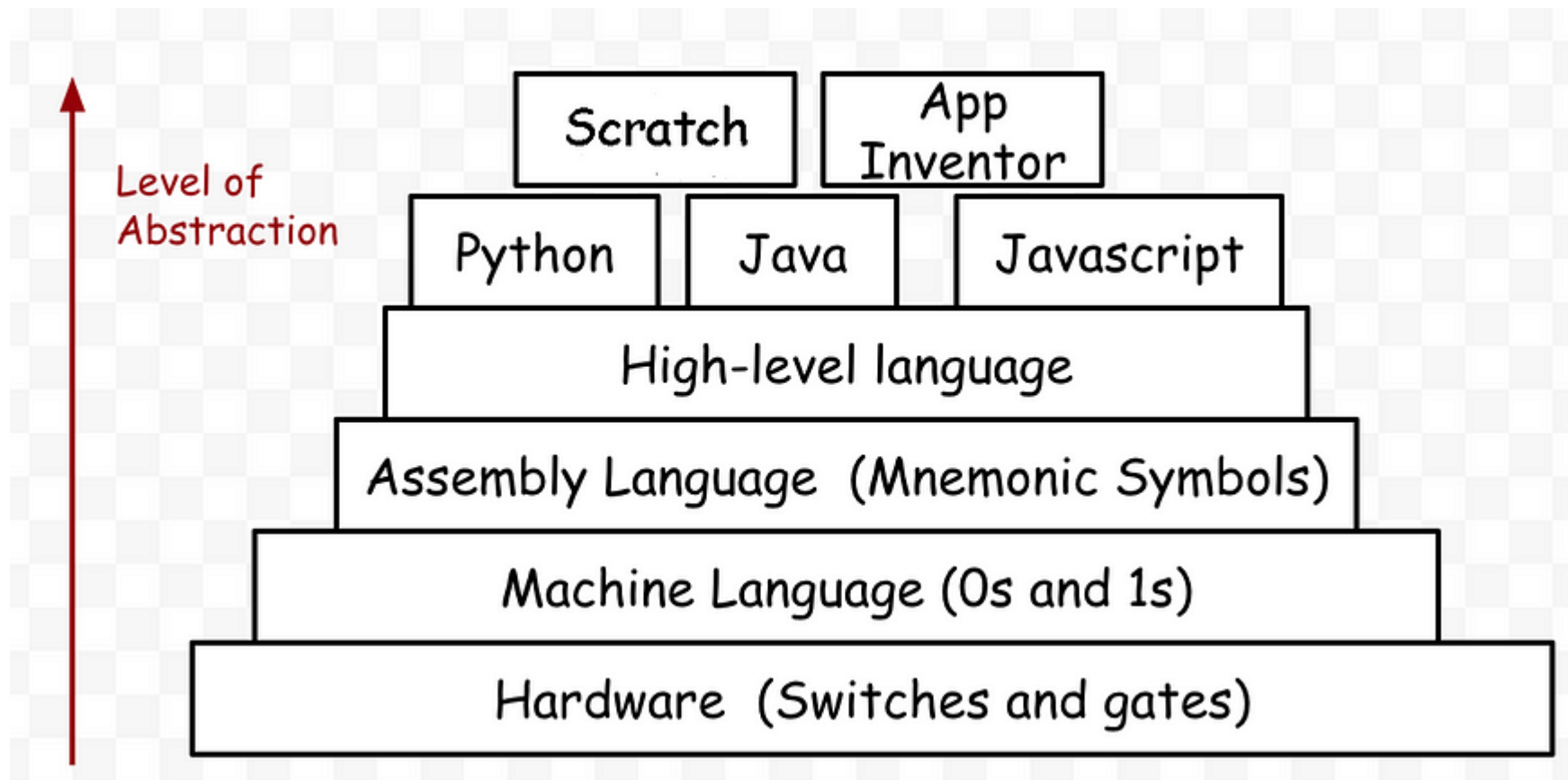


Học viện Công nghệ Bưu chính Viễn thông

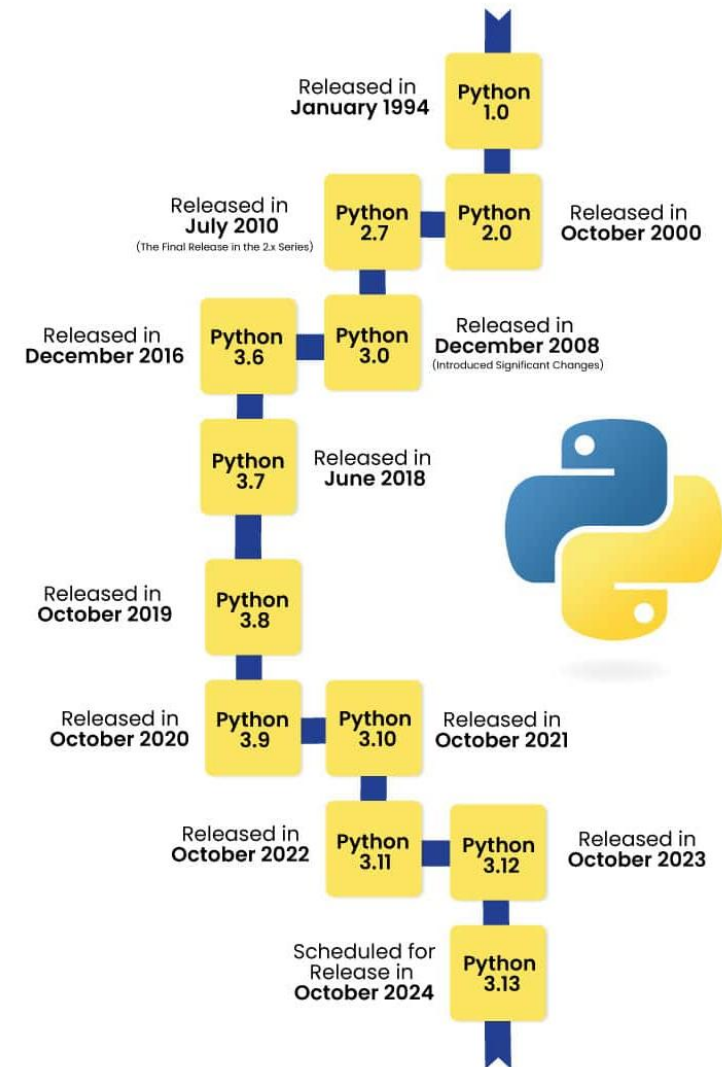
1. Giới thiệu về Python
2. Cài đặt Python và công cụ
3. Biến, kiểu dữ liệu, toán tử
4. Cấu trúc điều kiện và vòng lặp
5. Bài tập thực hành

1. Giới thiệu về Python

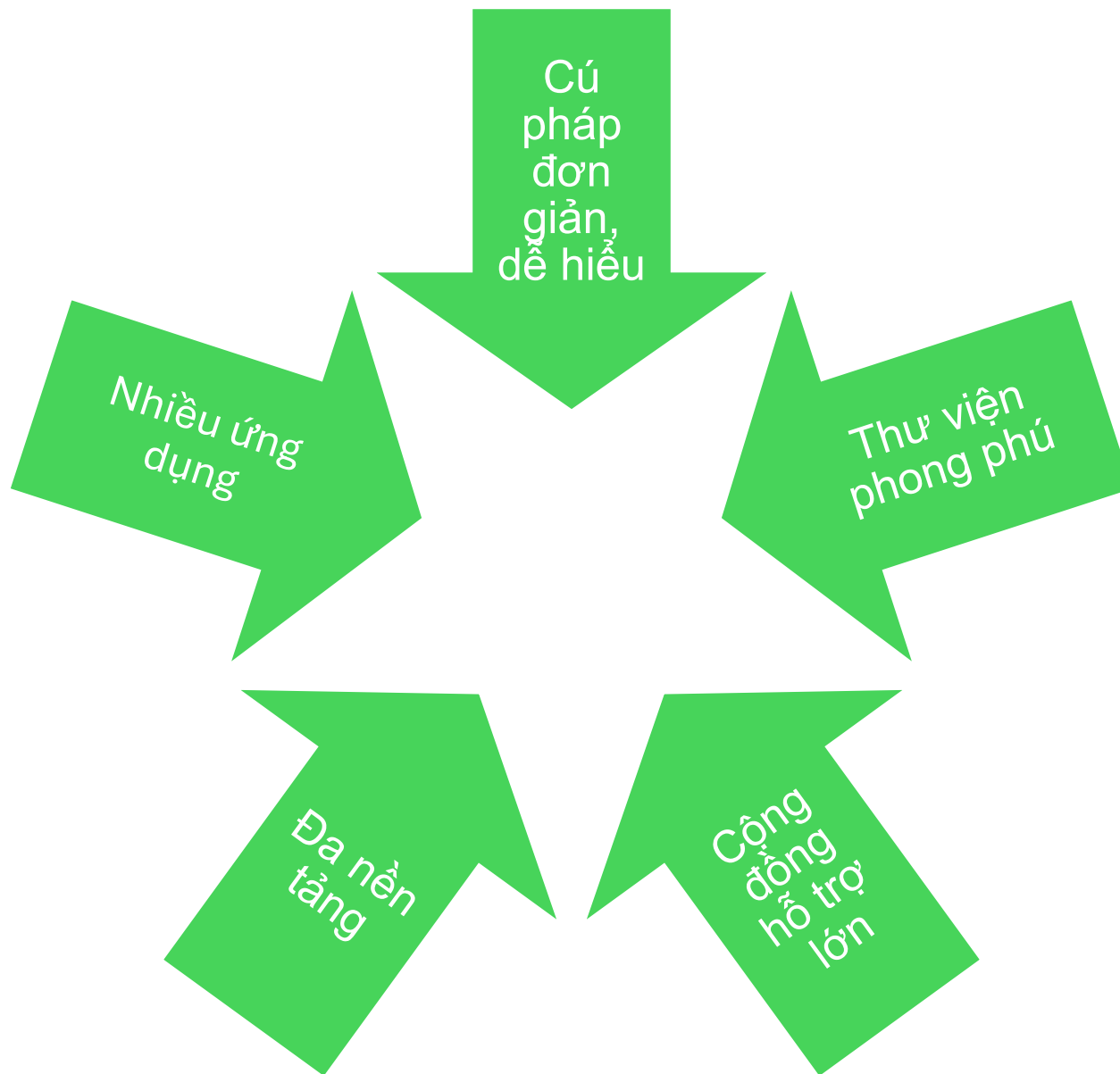
- ▶ Python là ngôn ngữ lập trình bậc cao



- ▶ Ra đời năm 1991
- ▶ Logo Python thể hiện tính chất của ngôn ngữ:
 - Con trăn đại diện cho sự linh hoạt và sức mạnh của Python
 - Hai con trăn đối xứng thể hiện sự cân bằng và dễ đọc
 - Xanh dương tượng trưng cho sự tin cậy và chuyên nghiệp
 - Vàng biểu thị sự sáng tạo, đổi mới, và thân thiện với người dùng



Tại sao lựa chọn Python?

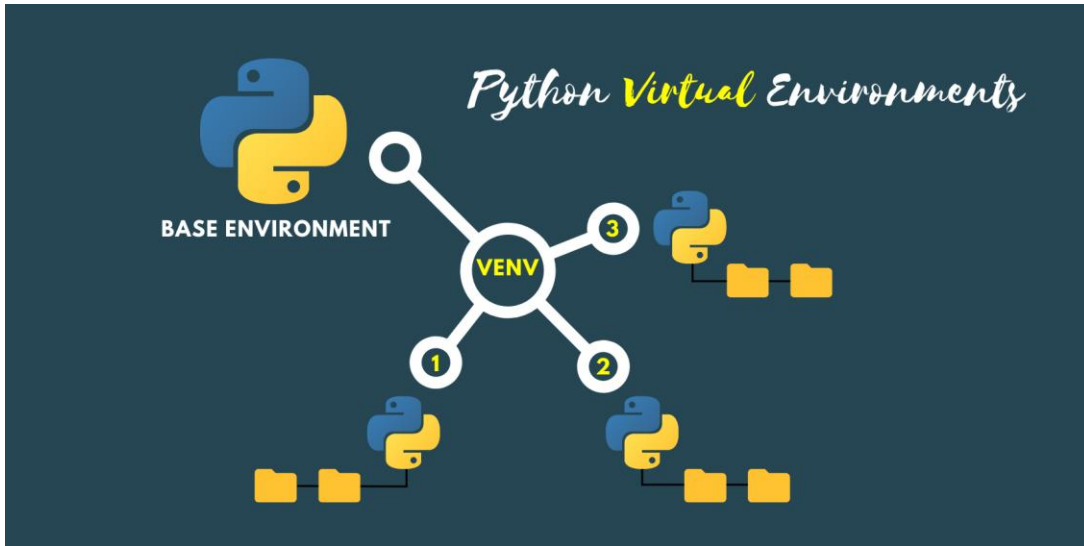


- ▶ Python được sử dụng rộng rãi trong nhiều lĩnh vực



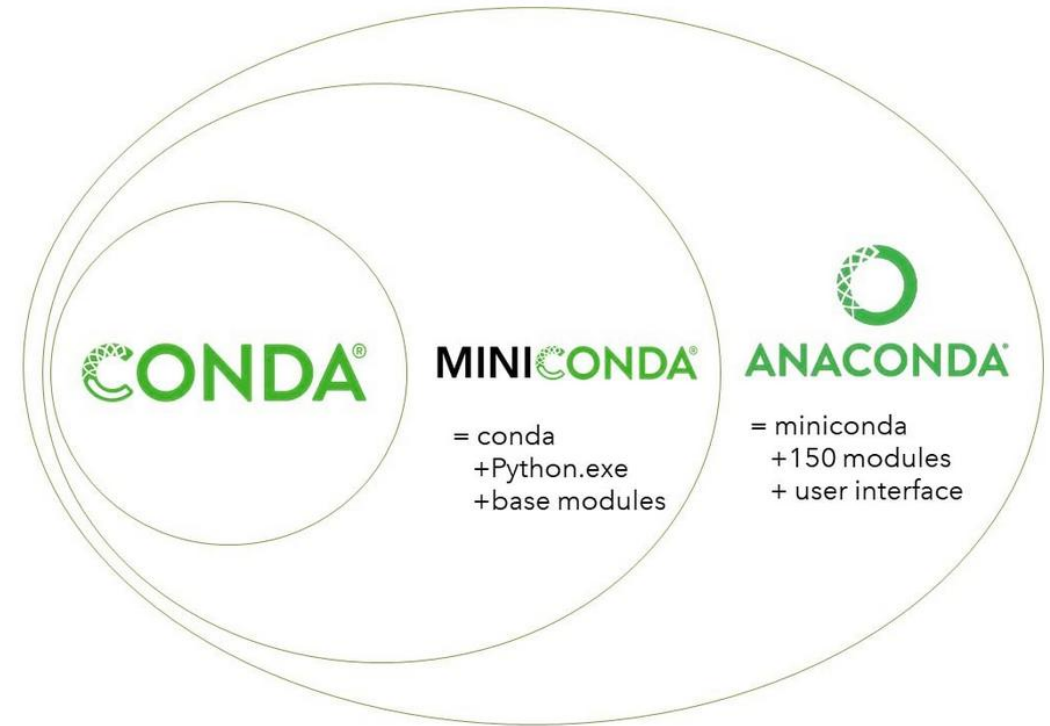
2. Cài đặt Python và công cụ

- ▶ Tải Python từ trang chủ: python.org
- ▶ Cài đặt như chương trình thông thường
- ▶ Đi kèm pip - công cụ quản lý thư viện Python
- ▶ Kiểm tra pip: `pip --version`



- ▶ venv (Virtual Environment) là công cụ tạo môi trường ảo trong Python
- ▶ Cô lập môi trường giúp tránh xung đột giữa các dự án
- ▶ Quản lý thư viện dễ dàng, không ảnh hưởng đến hệ thống Python toàn cục
- ▶ Hỗ trợ tái tạo môi trường nhanh chóng bằng tệp requirements.txt

- ▶ Là một phân phối Python phổ biến, hỗ trợ quản lý gói và môi trường ảo.
- ▶ Cung cấp công cụ [conda](#) để cài đặt, nâng cấp và quản lý thư viện
- ▶ Hỗ trợ tạo và quản lý môi trường ảo để làm việc với nhiều phiên bản thư viện khác nhau
- ▶ Tích hợp sẵn nhiều thư viện và công cụ
- ▶ Miniconda là phiên bản nhẹ hơn của Anaconda, bao gồm conda, Python và một số công cụ cơ bản



- ▶ Tải PyCharm từ trang chủ: [Pycharm](https://www.jetbrains.com/pycharm/)
- ▶ Là một IDE mạnh mẽ dành cho lập trình Python, phát triển bởi JetBrains
- ▶ Hỗ trợ nhiều công cụ lập trình: tự động hoàn thành mã, gỡ lỗi, kiểm tra lỗi cú pháp, ...
- ▶ Nhiều công cụ tiện ích tích hợp: terminal, version control, ...
- ▶ Tương thích cả venv và conda



3. Biến, kiểu dữ liệu, toán tử

- ▶ Là vùng nhớ để lưu trữ giá trị và có thể thay đổi trong quá trình chạy chương trình
- ▶ Python không yêu cầu khai báo kiểu dữ liệu trước khi sử dụng biến
- ▶ Cú pháp gán biến:

```
x = 10  
name = "Python"
```

- ▶ Quy tắc đặt tên biến:
 - Chỉ chứa chữ cái (a-z, A-Z), số (0-9) và dấu gạch dưới (_)
 - Không bắt đầu bằng số.
 - Không trùng với từ khóa Python (if, else, while, ...)

► Python hỗ trợ nhiều kiểu dữ liệu:

- Số nguyên (int): `x = 10`
- Số thực (float): `y = 3.14`
- Chuỗi (str): `name = "Python"`
- Boolean (bool): `is_valid = True`
- Danh sách (list): `numbers = [1, 2, 3, 4]`
- Tuple (tuple): `point = (1, 2)`
- Từ điển (dict): `student = {"name": "Alice", "age": 20}`

- ▶ Chuyển đổi giữa các kiểu dữ liệu trong Python:

```
x = 5  
y = str(x)  
z = float(x)
```

- ▶ Một số hàm chuyển đổi phổ biến:

- int() → Chuyển sang số nguyên
- float() → Chuyển sang số thực
- str() → Chuyển sang chuỗi
- list() → Chuyển sang danh sách
- tuple() → Chuyển sang tuple

- ▶ `input()` nhận dữ liệu từ người dùng dưới dạng chuỗi
- ▶ Cần ép kiểu nếu nhập số (`int()`, `float()`)
- ▶ `print()` in nhiều giá trị, mặc định cách nhau bởi dấu cách
- ▶ `sep="..."` thay đổi dấu phân cách giữa các giá trị
- ▶ `end="..."` thay đổi ký tự kết thúc (mặc định là xuống dòng `\n`)
- ▶ f-string giúp định dạng chuỗi một cách dễ dàng



```
# Input data from the keyboard
name = input("Enter your name: ") # Enter name (string)
age = int(input("Enter your age: ")) # Enter age and convert to an integer

# Output data to the screen
print("Hello,", name) # Print a normal string
print("You are", age, "years old") # Print multiple values

# Customize print()
print("Python", "Programming", sep=" - ") # Change the separator
print("Hello", end=" ") # Do not move to a new line
print("world!") # Continue on the same line


# Format string with f-string
print(f"{name} is {age} years old.")
```

► Toán tử là các ký hiệu dùng để thực hiện phép toán trên dữ liệu


► Các loại toán tử chính:

- Toán tử số học (+, -, *, /, %, //,)
- Toán tử gán (=, +=, -=, =, ...)
- Toán tử so sánh (==, !=, >, <, >=, <=)
- Toán tử logic (and, or, not)
- Toán tử bit (&, |, ^, ~, <<, >>)
- Toán tử membership (in, not in)
- Toán tử identity (is, is not)


Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+X, -X, ~X	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
		Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR



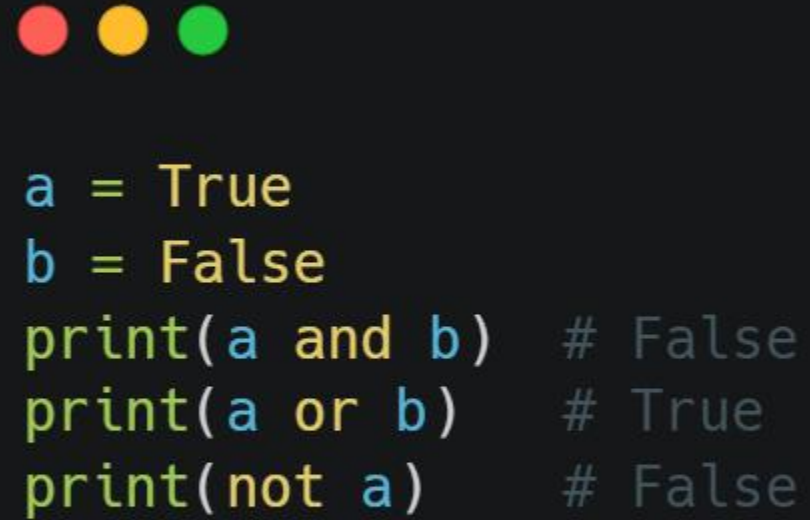
```
a = 10
b = 3
print(a + b)    # 13
print(a - b)    # 7
print(a * b)    # 30
print(a / b)    # 3.3333...
print(a // b)   # 3
print(a % b)    # 1
print(a ** b)   # 1000
```



```
x = 5
y = 10
print(x == y) # False
print(x != y) # True
print(x > y)  # False
print(x <= y) # True
```




```
x = 10
x += 5 # x = x + 5 → x = 15
x *= 2 # x = x * 2 → x = 30
x //= 3 # x = x // 3 → x = 10
```



```
a = True
b = False
print(a and b)    # False
print(a or b)     # True
print(not a)      # False
```



```
x = 5 # 101
y = 3 # 011
print(x & y) # 1 (001)
print(x | y) # 7 (111)
print(x ^ y) # 6 (110)
print(~x) # -6 (bitwise not x = -(x+1))
```

```
lst = [1, 2, 3, 4]
print(3 in lst) # True
print(5 not in lst) #
True
```

```
a = [1, 2, 3]
b = a
c = [1, 2, 3]
print(a is b)      # True
print(a is c)      # False
```

▶ Toán tử ternary giúp viết gọn biểu thức điều kiện if-else


▶ Cú pháp:

```
value_if_true if condition else value_if_false
```

▶ Ví dụ:

```
age = 18  
status = "Adult" if age >= 18 else "Minor"  
print(status) # Output: Adult
```

- ▶ Là các biến không thay đổi giá trị trong suốt chương trình
- ▶ Python không có từ khóa const, nhưng quy ước đặt tên hằng số bằng chữ IN HOA



```
PI = 3.14159  
GRAVITY = 9.8
```

4. Cấu trúc điều kiện và vòng lặp

▶ Python chuẩn quy định sử dụng 4 dấu cách (spaces) – hay Tab để thụt lề các khối lệnh

▶ Cú pháp:

```
if condition:
    # Execute if the condition is true
elif another_condition:
    # Execute if this condition is true
else:
    # Execute if none of the conditions are true
```

▶ Ví dụ:

```
age = 18
if age < 18:
    print("You are not old enough.")
elif age == 18:
    print("You are just old enough.")
else:
    print("You are old enough.")
```

- ▶ Có thể lồng nhiều câu lệnh if-else để kiểm tra điều kiện phức tạp
- ▶ Ví dụ:

```
score = 85
if score >= 50:
    if score >= 80:
        print("Excellent!")
    else:
        print("Pass")
else:
    print("Fail")
```

- ▶ match-case thay thế if-elif-else khi có nhiều trường hợp cụ thể
- ▶ Chỉ có từ Python 3.10
- ▶ Ví dụ:

```
day = "Monday"
match day:
    case "Monday":
        print("Start of the week!")
    case "Friday":
        print("Weekend is coming!")
    case _:
        print("Just a regular day.")
```




```
for i in range(5): # Iterate from 0 to 4
    print(i)
```

```
fruits = ["Apple", "Orange", "Banana"]
for fruit in fruits:
    print(fruit)
```

Vòng lặp while



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

► Dùng vòng lặp lồng nhau để duyệt nhiều mức dữ liệu

```
# Using nested for loops to create a multiplication table
for i in range(1, 4): # Outer loop (rows)
    for j in range(1, 4): # Inner loop (columns)
        print(i * j, end=" ") # Print the product
    print() # Move to the next line

# Using while inside a for loop
for i in range(1, 4): # Outer for loop
    j = 1 # Initialize j for inner while loop
    while j <= 3: # Inner while loop
        print(f"i={i}, j={j}") # Print values
        j += 1 # Increment j

# Using nested while loops to print a right-angled triangle
i = 1
while i <= 3: # Outer while loop
    j = 1
    while j <= i: # Inner while loop
        print("*", end=" ") # Print asterisks
        j += 1
    print() # Move to the next line
    i += 1 # Increment outer loop
```

- ▶ Lệnh break: Dừng vòng lặp ngay lập tức
- ▶ Lệnh continue: Bỏ qua phần còn lại của vòng lặp, tiếp tục lần lặp sau

```
for i in range(10):  
    if i == 5:  
        break # Stop the loop  
    print(i)  
  
for i in range(5):  
    if i == 2:  
        continue # Skip the value 2  
    print(i)
```

5. Bài tập thực hành

Cho số tự nhiên N không quá 9 chữ số. Hãy in ra giá trị bình phương của N .

Input

Dòng đầu ghi số bộ test. Mỗi bộ test có duy nhất một số tự nhiên không quá 9 chữ số.

Output

Với mỗi bộ test, ghi ra kết quả trên một dòng.

Ví dụ

Input	Output
2	1
1	529
23	

Cho số nguyên dương N không quá 9 chữ số. Hãy in ra giá trị thập phân $1/N$.

Input

Dòng đầu ghi số bộ test. Mỗi bộ test có duy nhất một số nguyên dương không quá 9 chữ số.

Output

Với mỗi bộ test, ghi ra kết quả trên một dòng với đúng 15 số sau dấu phẩy.

Ví dụ

Input	Output
2	1.000000000000000
1	0.043478260869565
23	

Hình vuông rỗng với dấu *

Viết chương trình nhập vào n (không quá 100) là cạnh của hình vuông và thực hiện in ra hình vuông rỗng các ký tự * theo mẫu trong ví dụ.

Input

Chỉ có một số nguyên dương N không quá 100.

Output

Ghi ra kết quả theo mẫu.

Ví dụ

Input	Output
4	**** * * .. * * .. ****

Viết chương trình kiểm tra một số nguyên dương có phải **số nguyên tố** hay không.

Input

Dòng đầu của dữ liệu vào ghi số bộ test. Mỗi dòng tiếp theo có một nguyên dương không quá 9 chữ số.

Output

Kết quả in ra YES nếu đó là số nguyên tố, in ra NO nếu ngược lại.

Ví dụ:

Input	Output
3	NO
123456	YES
997	NO
111111111	

Viết chương trình nhập vào một số n , không quá 10 chữ số.

Hãy thực hiện đếm số lần xuất hiện của các chữ số nguyên tố trong n và in ra màn hình. (Liệt kê theo thứ tự xuất hiện các chữ số)

Input

Chỉ có một số nguyên dương N không quá 10 chữ số.

Output

Ghi ra kết quả, mỗi dòng ghi một số nguyên tố và số lần xuất hiện theo thứ tự xuất hiện.

Ví dụ

Input	Output
112345	2 1
	3 1
	5 1