



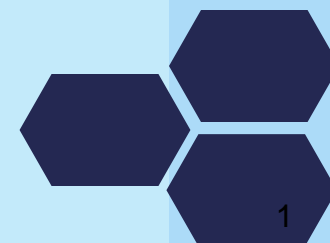
# Bài giảng

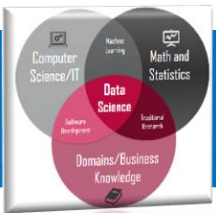
# NHẬP MÔN

# KHOA HỌC DỮ LIỆU

(Data Science)

ThS. Phạm Đình Tài  
0985.73.39.39  
pdtai@ntt.edu.vn

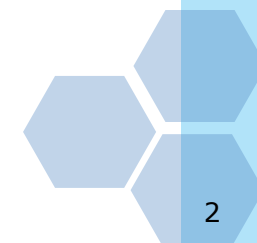


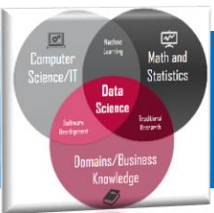


**CHƯƠNG**

**2**

# Ngôn ngữ Python

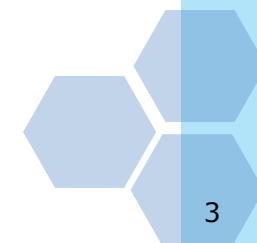




## **Bài 2.2**

# **Kiểu dữ liệu & phép toán**

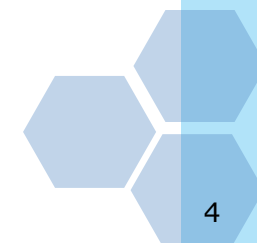
- 1. Kiểu dữ liệu và phép toán liên quan***
- 2. Cấu trúc rẽ nhánh***
- 3. Vòng lặp***
- 4. Hàm***
- 5. Bài tập***





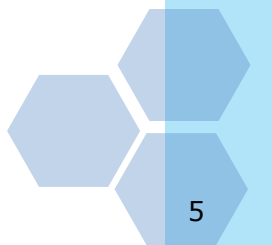
# Ôn lại

- Biến không cần khai báo trước, không cần chỉ kiểu
- Dữ liệu chuỗi nằm trong cặp nháy đơn ('), nháy kép ("), hoặc ba dấu nháy (""") – nếu viết nhiều dòng
- Sử dụng chuỗi thoát (escape sequence) để khai báo các ký tự đặc biệt
  - Sử dụng chuỗi “trần”: **r** **”nội dung”**
- Dùng dấu thăng (#) để viết dòng chú thích
- Dùng hàm **print** để in dữ liệu
- Dùng hàm **input** để nhập dữ liệu
  - Có thể kết hợp với hàm chuyển đổi kiểu





# Kiểu dữ liệu & phép toán liên quan





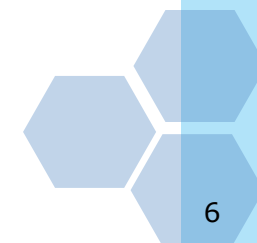
# Kiểu số

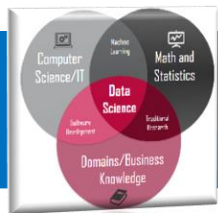
❖ Python viết số nguyên theo nhiều hệ cơ số

- `A = 1234` # hệ cơ số 10
- `B = 0xAF1` # hệ cơ số 16
- `C = 0o772` # hệ cơ số 8
- `D = 0b1001` # hệ cơ số 2

❖ Chuyển đổi từ số nguyên thành string ở các hệ cơ số khác nhau

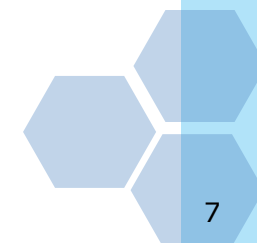
- `K = str(1234)` # chuyển thành str ở hệ cơ số 10
- `L = hex(1234)` # chuyển thành str ở hệ cơ số 16
- `M = oct(1234)` # chuyển thành str ở hệ cơ số 8
- `N = bin(1234)` # chuyển thành str ở hệ cơ số 2

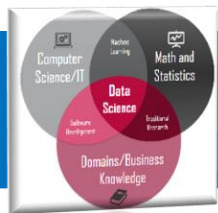




# Kiểu số

- ❖ Từ python 3, số nguyên không có giới hạn số chữ số
- ❖ Số thực (float) trong python có thể viết kiểu thông thường hoặc dạng khoa học
  - `x = 12.34`
  - `y = 314.15279e-2` # dạng số nguyên và phần mũ 10
- ❖ Python hỗ trợ kiểu số phức, với chữ j đại diện cho phần ảo
  - `A = 3+4j`
  - `B = 2-2j`
  - `print(A+B)` # sẽ in ra (5+2j)





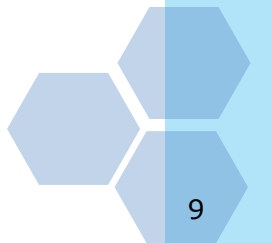
# Phép toán

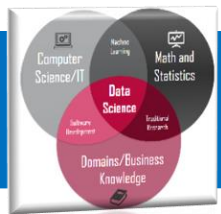
- ❖ Python hỗ trợ nhiều phép toán số, logic, so sánh và phép toán bit
  - Các phép toán số thông thường: `+`, `-`, `*`, `%`, `**`
  - Python có 2 phép chia:
    - Chia đúng (`/`): `10/3` # `3.3333333333333335`
    - Chia nguyên (`//`): `10/3` # `3` (nhanh hơn phép `/`)
  - Các phép logic: `and`, `or`, `not`
    - Python không có phép `xor` logic, trường hợp muốn tính phép xor thì thay bằng phép so sánh khác (`bool(a) != bool(b)`)
  - Các phép so sánh: `<`, `<=`, `>`, `>=`, `!=`, `==`
  - Các phép toán bit: `&`, `|`, `^`, `~`, `<<`, `>>`
  - Phép kiểm tra tập (`in`, `not in`): `1 in [1, 2, 3]`





# Cấu trúc rẽ nhánh





# Cấu trúc rẽ nhánh **if-else**

**if** expression:  
# if-block

**if** expression:  
# if-block

**elif** 2-expression:  
# 2-if-block

**elif** 3-expression:  
# 3-if-block

**elif** n-expression:  
# n-if-block

**if** expression:  
# if-block

**else:**  
#else-block

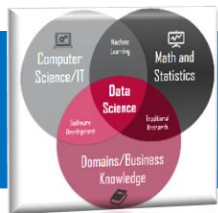
**if** expression:  
# if-block

**elif** 2-expression:  
# 2-if-block

...

**elif** n-expression:  
# n-if-block

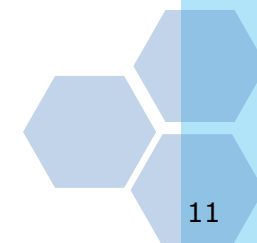
**else:**  
#else-block

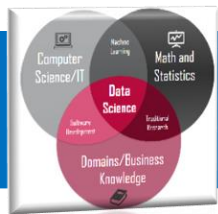


# Cấu trúc rẽ nhánh **if-else**

**Chú ý: python nhạy cảm với việc viết khối mã**

```
name = input("What's your name? ")
print("Nice to meet you " + name + "!")
age = int(input("Your age? "))
print("You are already", age, "years old,",
      name, "!")
if age>=18:
    print("Đủ tuổi đi bầu cử")
    if age>100:
        print("Có vẻ sai sai!")
else:
    print("Nhỏ quá")
```





# “phép toán” **if**

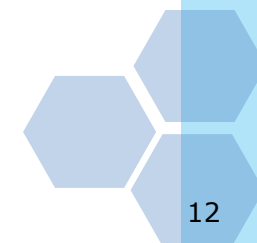
- Python có cách sử dụng **if** khá kì cục (theo cách nhìn của những người đã biết lệnh **if** trong một ngôn ngữ khác)
- Nhưng cách viết này rất hợp lý xét về mặt ngôn ngữ và cách đọc điều kiện logic

Cú pháp: **A if <điều-kiện> else B**

Giải thích: phép toán trả về A nếu điều-kiện là đúng, ngược lại trả về B

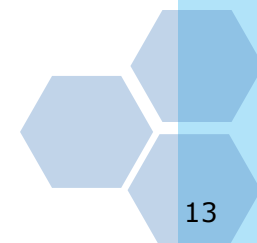
**Ví dụ:**

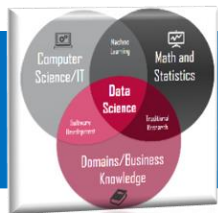
**X = A if A > B else B    # X là max của A và B**





# Vòng lặp





# Vòng lặp **while**

**while** expression:

# while-block

**while** expression:

# while-block-1

**continue**

# while-block-2

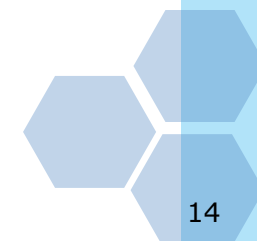
**while** expression:

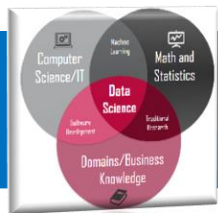
# while-block

**else:**

#else-block

- Lặp **while** trong python tương đối giống trong các ngôn ngữ khác
- Trong khối lệnh while (lệnh lặp nói chung) có thể dung **continue** hoặc **break** để về đầu hoặc cuối khối lệnh
- Khối “**else**” sẽ được thực hiện sau khi toàn bộ vòng lặp đã chạy xong
  - Khối này sẽ không chạy nếu vòng lặp bị “**break**”





# Vòng lặp **for**

```
for variable_1, variable_2, variable_n in sequence:
```

```
# for-block
```

```
for variable_1, variable_2, variable_n in sequence:
```

```
# for-block
```

```
else:
```

```
#else-block
```

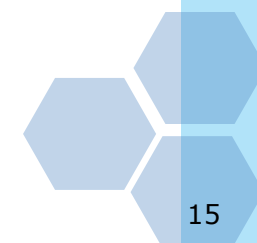
- Vòng lặp for sử dụng để duyệt danh sách, khối else làm việc tương tự như ở vòng lặp while
- Dùng hàm **range(a, b)** để tạo danh sách gồm các số từ a đến b-1, hoặc tổng quát hơn là **range(a, b, c)** trong đó c là bước nhảy

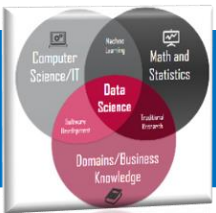
```
for d in range(10,20): # in các số từ 10 đến 19
```

```
print(d)
```

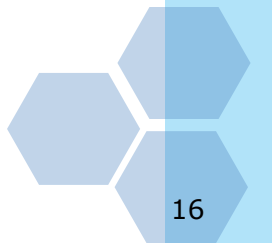
```
for d in range(20,10,-1): # in các số từ 20 đến 11
```

```
print(d)
```





# Hàm







# Hàm

- Cú pháp khai báo hàm rất đơn giản  
`def <tên-hàm> (danh-sách-tham-số) :`  
`<lệnh 1>`

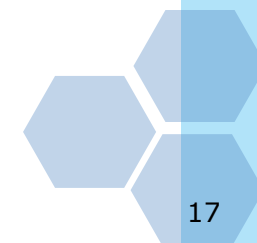
...

`<lệnh n>`

- Ví dụ: hàm tính tích 2 số

```
def tích(a, b):  
    return a*b
```

Hàm trả về kết quả bằng lệnh **return**, nếu không trả về thì coi như trả về **None**





# Hàm

- Hàm có thể chỉ ra giá trị mặc định của tham số

```
def tinh(a, b = 1):  
    return a*b
```

- Như vậy với hàm trên ta có thể gọi thực hiện nó:

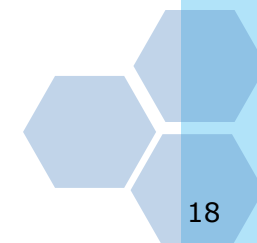
```
print(tinh(10, 20)) # 200
```

```
print(tinh(10)) # 10
```

```
print(tinh(a=5)) # 5
```

```
print(tinh(b=6, a=5)) # 30
```

- Chú ý: các tham số có giá trị mặc định phải đứng cuối danh sách tham số





# Thank You!

ThS. Phạm Đình Tài  
0985.73.39.39  
pdtai@ntt.edu.vn

