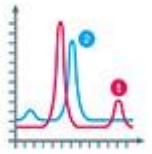
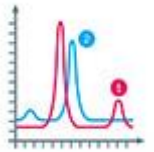


NHẬP MÔN LẬP TRÌNH KHOA HỌC DỮ LIỆU

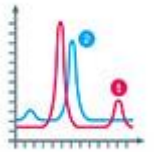
Bài 3: Ngôn Ngữ Lập Trình Python (2)

Nhắc lại kiến thức bài trước

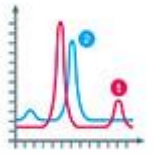


- Biến không cần khai báo trước, không cần chỉ kiểu
- Dữ liệu chuỗi nằm trong cặp nháy đơn ('), nháy kép ("), hoặc ba dấu nháy (""") – nếu viết nhiều dòng
  - Sử dụng chuỗi thoát (escape sequence) để khai báo các ký tự đặc biệt
  - Sử dụng chuỗi "trần": r"nội dung"
- Dùng dấu thăng (#) để viết dòng chú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

Nội dung



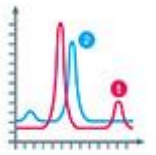
1. Kiểu dữ liệu và phép toán liên quan
2. Cấu trúc rẽ nhánh
3. Vòng lặp
4. Hàm
5. Bài tập



Phần 1

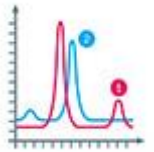
Kiểu dữ liệu và phép toán liên quan

Kiểu số



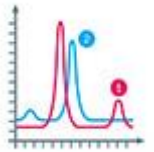
- Python viết số nguyên theo nhiều hệ cơ số
  - `A = 1234` `# hệ cơ số 10`
  - `B = 0xAF1` `# hệ cơ số 16`
  - `C = 0o772` `# hệ cơ số 8`
  - `D = 0b1001` `# hệ cơ số 2`
- Chuyển đổi từ số nguyên thành string ở các hệ cơ số khác nhau
  - `K = str(1234)` `# chuyển thành str ở hệ cơ số 10`
  - `L = hex(1234)` `# 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

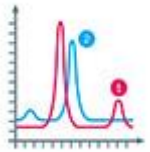


- Từ python 3, số nguyên không có giới hạn số chữ số
- Số thực (float) trong python có thể viết kiểu thông thường hoặc dạng khoa học
  - `X = 12.34`
  - `Y = 314.15279e-2  # dạng số nguyên và phần mũ 10`
- Python hỗ trợ kiểu số phức, với chữ j đại diện cho phần ảo
  - `A = 3+4j`
  - `B = 2-2j`
  - `print(A+B)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

Phép toán



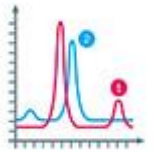
- Python hỗ trợ nhiều phép toán số, logic, so sánh và phép toán bit
  - Các phép toán số thông thường: `+, -, *, %, **`
  - Python có 2 phép chia:
    - Chia đúng (/): `10/3` `# 3.3333333333333335`
    - Chia nguyên (//): `10/3` `# 3 (nhanh hơn phép /)`
  - Các phép logic: `and, or, not`
    - Python 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]`



Phần 2

Cấu trúc rễ nhánh

Cấu trúc rẽ nhánh if-else



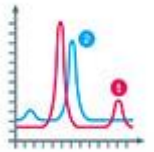
```
if expression:
    # If-block
```

```
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
```

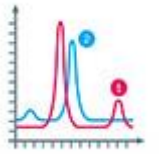
Chú ý khối mã trong if-else



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

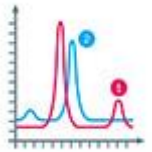
```
name = input("What's your name? ")
print("Nice to meet you " + name + "!")
age = int(input("Your age? "))
print("You are already", age, "years old,", name, "!")
if age>=18:
    print("Đủ tuổi đi bầu")
    if age>100:
        print("Có vẻ sai sai!")
else:
    print("Nhỏ quá")
```

"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:

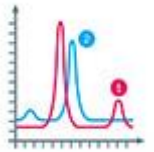
```
X = A if A > B else B          # X là max của A và B
```



Phần 3

Vòng lặp

Vòng lặp while và for



```
while expression:
    # while-block
```

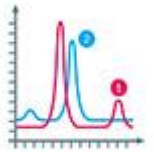
```
while expression:
    #while-block-1
    continue
    #while-block-2
```

```
while expression:
    # while-block
else:
    # else-block
```

```
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 while



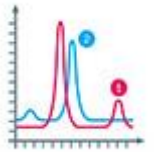
```
while expression:
    # while-block
```

```
while expression:
    #while-block-1
    continue
    #while-block-2
```

```
while expression:
    # while-block
else:
    # else-block
```

- Chú ý:
  - Lặp while trong python tương đối giống trong các ngôn ngữ khác
  - Trong khối lệnh while (lệnh lặp nói chung) có thể dùng 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

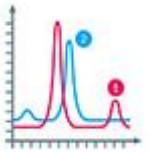


```
for variable_1, variable_2, .. variable_n in sequence:
    # for-block
```

```
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

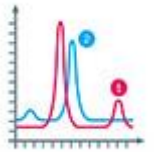
```
for d in range(10,20):          # in các số từ 10 đến 19
    print(d)
for d in range(20,10,-1):       # in các số từ 20 đến 11
    print(d)
```



Phần 4

Hàm

Hàm



- Cú pháp khai báo hàm rất đơn giản

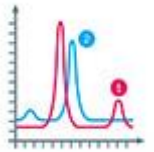
```
def <tên-hàm>(danh-sách-tham-số):
    <lệnh 1>
    …
    <lệnh n>
```

- Ví dụ: hàm tính tích 2 số

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

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

Hàm



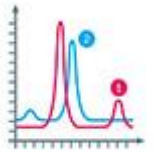
- Hàm có thể chỉ ra giá trị mặc định của tham số

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

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

```
print(tich(10, 20))     # 200
print(tich(10))         # 10
print(tich(a=5))        # 5
print(tich(b=6, a=5))   # 30
```

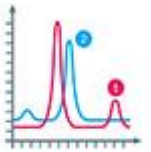
- Chú ý: các tham số có giá trị mặc định phải đứng cuối danh sách tham số



Phần 5

Bài tập

Bài tập



1. Viết chương trình nhập số A và kiểm tra xem A có phải là số nguyên tố hay không?
2. Viết chương trình nhập hai số A và B, in ra tất cả các số nguyên tố nằm trong khoảng [A, B].
3. Nhập 2 số A và B, tính và in ra màn hình ước số chung lớn nhất và bội số chung nhỏ nhất của hai số đó.
4. Nhập tọa độ 3 điểm A, B và C trên mặt phẳng 2 chiều. Hãy kiểm tra và chỉ ra hình dạng của tam giác ABC (đều, vuông, cân, vuông cân, tù, nhọn,...)