NHẬP MÔN
TƯ DUY TÍNH TOÁN

Bài 3: Cấu trúc điều khiển trong python

Tóm tắt nội dung bài trước

- Hai cách thực thi python: chạy chương trình và dòng lệnh
- Dùng dấu thăng (#) để viết dòng chú thích
- Biến không cần khai báo trước, không cần chỉ kiểu
- Dữ liệu chuỗi nằm trong cặp nháy đơn ('), nháy kép ("), hoặc ba dấu nháy kép (""") – nếu viết nhiều dòng
  - Sử dụng chuỗi thoát (\)để khai báo các ký tự đặc biệt
  - Sử dụng chuỗi thô: r"nội dung"
- Hàm print để in dữ liệu, hàm input để nhập dữ liệu
  - Có thể kết hợp với hàm chuyển đổi kiểu
- Kiểu số và phép toán có một số điểm cần chú ý
  - Số nguyên không giới hạn độ lớn
  - Phép chia nguyên và phép chia chính xác

Nội dung

1. Phép toán “if”
2. Rẽ nhánh
3. Vòng lặp “while”
4. Vòng lặp “for”
5. Hàm
6. Bài tập

Phần 1

Phép toán “if”

Phép toán “if”

```
# X là max của A và B
X = A if A > B else B
# N có phải là số nguyên tố có 1 chữ số hay không
A = "Đúng" if N in [2, 3, 5, 7] else "Sai"
# In ra màn hình “chẵn” nếu n chia hết cho 2,
#   in ra “lẻ” nếu ngược lại
print('chẵn' if (n % 2) == 0 else "lẻ")
# Sinh viên có được thi hay không?
print("được thi" if so_buoi_nghi < 3 else "không được thi")
# Biện luận nghiệm phương trình bậc 2 (if lồng nhau)
KQ = "một nghiệm" if delta == 0 else \
                    "vô nghiệm" if delta < 0 else "hai nghiệm"
```

Phép toán "if"

- Cú pháp: `A if <điều-kiện> else B`
- Thực hiện:
  - Phép toán trả về A nếu <điều-kiện> là đúng, ngược lại trả về B
  - A và B có thể là các giá trị, biểu thức tính toán, lời gọi hàm,…
  - Các phép toán if cũng có thể lồng nhau
- Cách sử dụng if này khá kì cục, nhưng hợp lý nếu xét về mặt ngôn ngữ và cách đọc điều kiện logic
- Bài tập: Biến X để lưu tình trạng gửi SMS, X=0 tức là chưa gửi được, X=1 tức là đã gửi thành công, X=2 tức là đã gửi và người nhận đã đọc. Viết câu lệnh sử dụng phép toán if để in ra màn hình thông báo tương ứng với giá trị của X.

Phần 2

Rẽ nhánh

Rẽ nhánh

```
# In thông báo nếu được điểm số loại giỏi
if diem >= 8:
    print("Chúc mừng bạn đã được điểm giỏi")
# In thông báo xem n chẵn hay lẻ
if (N % 2) == 0:
    print("N là số chẵn")
else:
    print("N là số lẻ")
# Biện luận nghiệm của phương trình bậc 2
if delta == 0:
    print("Phương trình có nghiệm kép")
elif delta < 0:
    print("Phương trình vô nghiệm")
else:
    print("Phương trình có hai nghiệm phân biệt")
```

Rẽ nhánh

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

```
if expression:
    # If-block
else:
    # else-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
elif 2-expression:
    # 2-if-block
...
elif n-expression:
    # n-if-block
else:
    # else-block
```

Rẽ nhánh

- Python chỉ có một cấu trúc rẽ nhánh duy nhất, sử dụng để lựa chọn làm một trong số nhiều công việc
  - Nhiều ngôn ngữ lập trình khác sử dụng if cho trường hợp 2 lối rẽ nhánh và switch cho trường hợp nhiều lối rẽ nhánh
- Nguyên tắc với rẽ nhánh if-elif-else:
  - Biểu thức điều kiện của if và elif phải có kết quả logic
  - Hệ thống sẽ lần lượt tính giá trị từng biểu thức điều kiện từ trên xuống dưới, bắt đầu từ phát biểu if
  - Nếu biểu thức điều kiện nào đúng thì khối lệnh tương ứng được thực hiện và bỏ qua các khối lệnh khác
  - Trường hợp mọi biểu thức điều kiện đều sai, khối lệnh ứng với else sẽ được thực hiện
  - Khối else là tùy chọn, không nhất thiết phải xuất hiện

Phần 3

Vòng lặp “while”

Vòng lặp while

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

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

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

- Vòng while thực hiện lặp lại khối lệnh chừng nào biểu thức điều kiện còn đúng
- Phát biểu continue trong khối lệnh sẽ ngắt khối lệnh hiện tại và bắt đầu một vòng lặp mới
- Phát biểu break sẽ kết thực vòng lặp ngay lập tức
- Khối else sẽ được thực hiện sau khi toàn bộ vòng lặp đã chạy xong, không bắt buộc phải có khối này
  - Khối này sẽ không chạy nếu vòng lặp bị "break"

Vòng lặp while

```
# In ra các số tự nhiên chia hết cho 7 nhỏ hơn 1000
n = 0
while n < 1000:
    if (n % 7) == 0:
        print(n)
    n += 1
# Tính tổng các số nhỏ hơn 1000 và không chia hết cho 3
t = 0
n = 0
while n < 1000:
    if (n % 3) != 0:
        t = t + n
    n += 1
print(t)
```

Vòng lặp while

```
# Với 10 triệu, gửi ngân hàng với lãi suất 5,1% hàng năm
#     tính xem sau bao nhiêu năm thì có ít nhất 50 triệu
so_tien = 10000000
lai_suat = 5.1/100
so_nam = 0

while so_tien < 50000000:
    so_nam += 1
    so_tien = so_tien * (1 + lai_suat)
    print("Số tiền sau", so_nam, "năm:", so_tien)

print("Sau", so_nam, "bạn sẽ có ít nhất 50 triệu.")
```

Vòng lặp while

```
# Ví dụ về lặp while có dùng break và else
# Nhập số n và kiểm tra xem nó có phải số nguyên tố hay không
n = int(input("Nhập số N: "))
x = 2
while x < n:
     if (n % x) == 0:
          print("N không phải số nguyên tố")
          break;
     x = x + 1
else:
     print("N là số nguyên tố")
```

Phần 4

Vòng lặp “for”

Vòng lặp for

- Cú pháp:

```
for <biến> in <danh-sách>:
    # khối for
else
    # khối else
```

- Vòng for cho phép sử dụng một <biến> lần lượt duyệt các giá trị trong <danh-sách>
- Tương tự như while, có thể sử dụng break và continue
- Khối else thực hiện sau khi toàn bộ vòng lặp đã chạy xong
  - Khối này sẽ không chạy nếu vòng lặp bị "break"
  - Không bắt buộc phải có khối này
  - Cách làm việc tương tự như ở vòng lặp while

Vòng lặp for

```
X = ['chó', 'mèo', 'lợn', 'gà']
# In ra các loài vật trong danh sách
for w in X:
    print(w)
# In ra các loại vật, ngoại từ loài ‘mèo’
for x in X:
    if x == 'mèo':
        continue
    print(x)
# In ra các loại vật, nếu gặp loài ‘mèo’ thì dừng luôn
for z in X:
    if z == 'mèo':
        break
    print(z)
```

Vòng lặp for

```
# Trường hợp một khoảng số khá lớn, không thể liệt kê được
# Ta sử dụng hàm range để tạo ra khoảng số
# In các số từ 10 đến 19: khoảng 10 đến 20, bước nhảy 1
for d in range(10, 20):
    print(d)
# In các số từ 20 đến 11: khoảng 20 đến 10, bước nhảy -1
for d in range(20, 10, -1):
    print(d)
# In các số lẻ từ 1 đến 100: khoảng 1 đến 100, bước nhảy 2
for d in range(1, 101, 2):
    print(d)
```

Phần 5

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 với tham số mặc định

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

```
# nếu không nói gì thì mặc định b=1
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 6

Bài tập

Bài tập

1. Viết chương trình cho phép người dùng nhập vào một dãy số tự nhiên (không biết trước độ dài), việc nhập dãy sẽ kết thúc khi người dùng nhập một số âm nào đó.
2. Viết hàm is_prime kiểm tra xem N có phải là số nguyên tố hay không?
3. Viết chương trình nhập hai số A và B, in ra tất cả các số nguyên tố nằm trong khoảng [A, B].
4. 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.
5. 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,...)