Lập trình Python

Nguyễn Hải Châu
Email: chaunh@vnu.edu.vn

Trường Đại học Công nghệ
Đại học Quốc gia Hà Nội

September 17, 2023

Biến và mảng (Variables and arrays)

Biến


```
var1 = 1
var2 = "This is a string variable"
print(var1, var2)

## 1 This is a string variable
```

Mảng (1)


```
arr1 = [6, 3, 1, 9, 9, 11]
arr2 = ["one", "two", "three"]
print(arr1)

## [6, 3, 1, 9, 9, 11]

print(arr2)

## ['one', 'two', 'three']

print(arr1[0]) # First element

## 6

print(arr1[2]) # Third element

## 1

print(arr1[-1], arr1[len(arr1)-1]) # Last element

## 11 11
```

Mảng (1a)

Mảng con (sub-array):

```
arr1 = [6, 3, 1, 9, 9, 11]
ll = len(arr1)
print(ll)

## 6

print(arr1[0:ll]) # toàn bộ mảng, từ phần tử 0 đến ll-1

## [6, 3, 1, 9, 9, 11]

print(arr1[0:])  # toàn bộ mảng, từ phần tử 0 đến ll-1 (cách
viết rút gọn)

## [6, 3, 1, 9, 9, 11]

print(arr1[1:ll]) # mảng con, lấy từ phần tử 1 đến ll-1

## [3, 1, 9, 9, 11]

print(arr1[2:4]) # mảng con, lấy từ phần tử 2 đến 3 (=4-1)

## [1, 9]
```

Mảng (2)


```
arr1 = [6, 3, 1, 9, 9, 11]
print(len(arr1)) # Number of elements in the array

## 6

arr1.remove(9) # Remove the first occurrence of 9 in the array
print(arr1)

## [6, 3, 1, 9, 11]

arr1.remove(arr1[1]) # Remove the second element of the array
print(arr1)

## [6, 1, 9, 11]

arr1.append(13) # Append one element to the array
print(arr1)

## [6, 1, 9, 11, 13]
```

Mảng (3)


```
arr1 = [6, 3, 1, 9, 9, 11]
for el in arr1: # print elements of the array one-by-one
    print(el)

## 6
## 3
## 1
## 9
## 9
## 11

for i in range(len(arr1)): # print elements by indices
    print(arr1[i])

## 6
## 3
## 1
## 9
## 9
## 11
```

Mảng (4)


```
arr1 = [6, 3, 1, 9, 9, 11]
arr1.sort() # sort the array ascendingly
print(arr1)

## [1, 3, 6, 9, 9, 11]

arr1.sort(reverse=True) # sort the array descendingly
print(arr1)

## [11, 9, 9, 6, 3, 1]
```

Các phép toán

Các phép toán số học


```
a = 5
b = 3
a+b, a-b # Cộng, trừ

## (8, 2)

a*b, a/b # Nhân chia

## (15, 1.6666666666666667)

a//b, a%b # Chia nguyên và tìm phần dư

## (1, 2)

a**3, a**(1/3) # Lũy thừa và căn

## (125, 1.7099759466766968)

5**2, 25**(1/2)

## (25, 5.0)
```

Các phép toán với chuỗi ký tự (1)


```
mystr = "Python is a programming language"
print(mystr[12:]) # print from 12th character

## programming language

sp1 = mystr.split() # split by space character
print(sp1) # sp1 is an array

## ['Python', 'is', 'a', 'programming', 'language']

sp2 = mystr.split("g") # split by "g" character
print(sp2) # sp2 is an array

## ['Python is a pro', 'rammin', ' lan', 'ua', 'e']
```

Các phép toán với chuỗi ký tự (2)


```
mystr = "Python is a programming language"
search = "gram"
if search in mystr:
    print(search, "is found")
else:
    print(search, "is not found")

## gram is found

search = "Gram"
if search in mystr:
    print(search, "is found")
else:
    print(search, "is not found")

## Gram is not found
```

Các phép toán với chuỗi ký tự (3)


```
strn1 = "2"
print(int(strn1)+1) # convert string to integer

## 3

strn2 = "4.1"
print(float(strn2)+1) # convert string to float

## 5.1

strn1 + strn2 # string concatenation

## '24.1'

int(strn1) + float(strn2) # not string concatenation

## 6.1

var = 3.5
print(str(var)) # convert number to string

## 3.5
```

Các cấu trúc điều khiển

Cấu trúc điều khiển for (1)


```
for i in range(3,6): # loop from 3 to 5
    print(i)

## 3
## 4
## 5

for i in range(3): # loop from 0 to 2
    print(i)

## 0
## 1
## 2
```

Cấu trúc điều khiển for (2)


```
arr1 = [3, 5, 2]
for i in range(len(arr1)): # loop for every element using index
    print(arr1[i])

## 3
## 5
## 2

for el in arr1: # loop for every element
    print(el)

## 3
## 5
## 2
```

Cấu trúc điều khiển while


```
x = 10
while x > 2: # unknown number of loops
    x = x / 1.5
    print("x =", x)

## x = 6.666666666666667
## x = 4.444444444444445
## x = 2.962962962962632
## x = 1.9753086419753088

print("Stopped")

## Stopped
```

Cấu trúc điều khiển if ... else


```
x = 10
if x > 5:
    print(x, "is large")
else:
    print(x, "is not large")

## 10 is large

x = 4
if x > 5:
    print(x, "is large")
else:
    print(x, "is not large")

## 4 is not large
```

Cấu trúc điều khiển if ... elif ... else (1)


```
x = 10
if x > 5:
    print(x, "is large")
elif x > 2:
    print(x, "is not large")
else:
    print(x, "is tiny")

## 10 is large

x = 4
if x > 5:
    print(x, "is large")
elif x > 2:
    print(x, "is not large")
else:
    print(x, "is tiny")

## 4 is not large
```

Cấu trúc điều khiển if ... elif ... else (2)


```
x = 1.5
if x > 5:
    print(x, "is large")
elif x > 2:
    print(x, "is not large")
else:
    print(x, "is tiny")

## 1.5 is tiny
```

Hàm (functions)

Định nghĩa và sử dụng hàm

Hàm là một khối lệnh có thể có tham số, có thể tái sử dụng và chỉ thực hiện khi được gọi đến

```
import math # sử dụng thư viện math

# Hàm tính cạnh huyền của một tam giác vuông biết hai cạnh góc
vuông
def hypotenuse(a, b): # Định nghĩa hàm
    if a<=0 or b<=0:
        print("ERROR: Edges of a triangle must be positive")
        return -1 # -1 means there are errors
    else:
        hp = math.sqrt(a**2+b**2)
        return hp

hypotenuse(3, 4) # Gọi (sử dụng) hàm

## 5.0

hypotenuse(0, 1)

## ERROR: Edges of a triangle must be positive
## -1
```

Hàm đệ qui

Một hàm có thể gọi đến chính nó, đó là hàm đệ qui (recursive function)

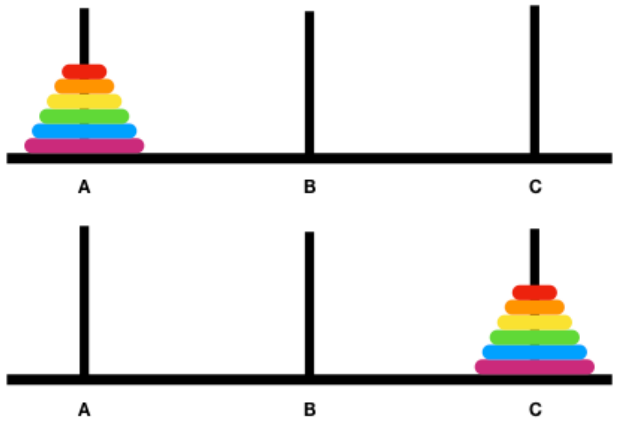
```
# Các hàm tính giai thừa
def factorial(n): # hàm không đệ qui
    if n == 0: # 0! = 1
        return 1
    else: # n! = 1.2.3..n
        ff = 1
        for i in range(2,n+1):
            ff = ff*i
        return ff
def factorial_r(n): # hàm đệ qui
    if n == 0: # 0! = 1
        return 1
    else: # n! = n.(n-1)!
        return n*factorial_r(n-1)
factorial(5)

## 120

factorial_r(5)

## 120
```

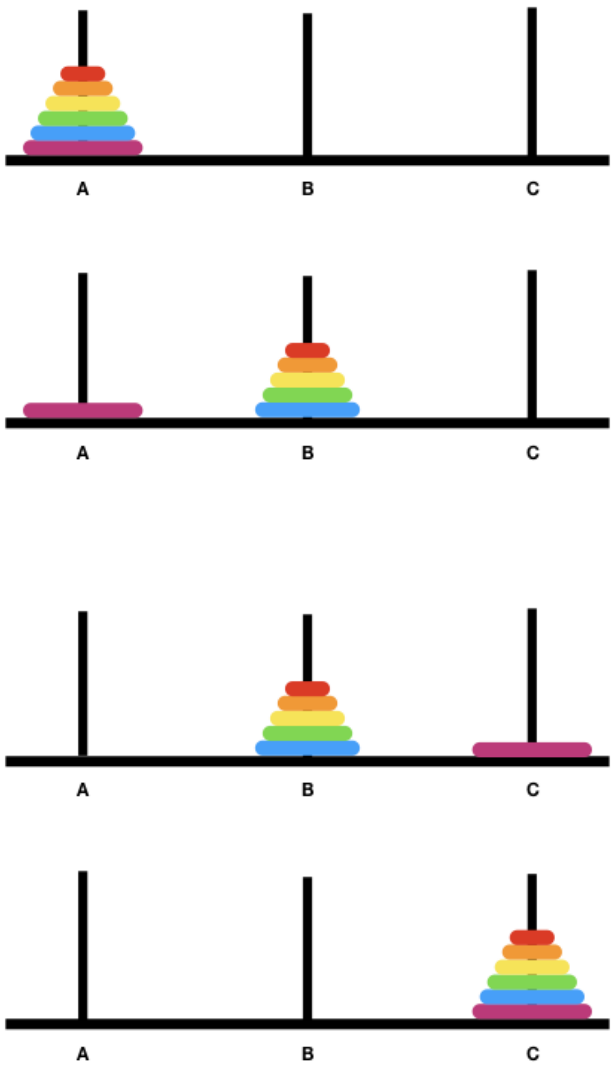
Ví dụ hàm đệ qui: bài toán tháp Hà Nội



Yêu cầu:

- Chuyển tất cả các đĩa từ A sang C, với B là trung gian
- Mỗi lần chỉ được chuyển 1 đĩa
- Đĩa to luôn nằm dưới đĩa bé

Cách chuyển



Chương trình Python (đệ qui)


```
# Chuyển n đĩa từ A sang C với B là trung gian
def HanoiTower(n, A, C, B):
    if n <= 0:
        print("ERROR: Số đĩa phải là số tự nhiên")
        return -1 # có lỗi
    elif n == 1:
        print("Chuyển đĩa", n, "từ", A, "sang", C)
        return 1 # Có 1 bước chuyển
    else:
        n1 = HanoiTower(n-1, A, B, C)
        print("Chuyển đĩa", n, "từ", A, "sang", C)
        n2 = HanoiTower(n-1, B, C, A)
        return n1+1+n2 # số bước chuyển
```

```
steps = HanoiTower(0, "A", "C", "B")

## ERROR: Số đĩa phải là số tự nhiên

print("Số lần chuyển:", steps)

## Số lần chuyển: -1
```

Chuyển 1, 3 đĩa từ A sang C, B là trung .


```
steps = HanoiTower(1, "A", "C", "B")

## Chuyển đĩa 1 từ A sang C

print("Số lần chuyển:", steps)

## Số lần chuyển: 1
```

```
steps = HanoiTower(3, "A", "C", "B")

## Chuyển đĩa 1 từ A sang C
## Chuyển đĩa 2 từ A sang B
## Chuyển đĩa 1 từ C sang B
## Chuyển đĩa 3 từ A sang C
## Chuyển đĩa 1 từ B sang A
## Chuyển đĩa 2 từ B sang C
## Chuyển đĩa 1 từ A sang C

print("Số lần chuyển:", steps)

## Số lần chuyển: 7
```

Chuyển 4 đĩa từ A sang C, B là trung gian


```
steps = HanoiTower(4, "A", "C", "B")

## Chuyển đĩa 1 từ A sang B
## Chuyển đĩa 2 từ A sang C
## Chuyển đĩa 1 từ B sang C
## Chuyển đĩa 3 từ A sang B
## Chuyển đĩa 1 từ C sang A
## Chuyển đĩa 2 từ C sang B
## Chuyển đĩa 1 từ A sang B
## Chuyển đĩa 4 từ A sang C
## Chuyển đĩa 1 từ B sang C
## Chuyển đĩa 2 từ B sang A
## Chuyển đĩa 1 từ C sang A
## Chuyển đĩa 3 từ B sang C
## Chuyển đĩa 1 từ A sang B
## Chuyển đĩa 2 từ A sang C
## Chuyển đĩa 1 từ B sang C

print("Số lần chuyển:", steps)

## Số lần chuyển: 15
```

Các phép toán logic

- Python có các phép toán logic `not`, `and`, `or`
- Phép toán 1 ngôi: `not`
- Phép toán 2 ngôi: `and`, `or`
- Các phép toán logic cho kết quả là một trong các giá trị `True`, `False`

Các phép toán logic

Phép toán **and**:

and	True	False
True	True	False
False	False	False

Phép toán **or**:

or	True	False
True	True	True
False	True	False

Phép toán **not**:

$$\text{not False} = \text{True}$$
$$\text{not True} = \text{False}$$

Các phép toán logic trong Python: not


```
# Phép toán not
print(not True)

## False

print(not False)

## True
```

Các phép toán logic trong Python: and


```
# Phép toán and
print(True and True)

## True

print(True and False)

## False

print(False and True)

## False

print(False and False)

## False
```

Các phép toán logic trong Python: or


```
# Phép toán or
print(True or True)

## True

print(True or False)

## True

print(False or True)

## True

print(False or False)

## False
```