Ôn tập Python

Th.S. Nguyễn Thành An
Trường ĐH Khoa học Tự nhiên,
Đại học Quốc gia - TP. HCM
ntan@selab.hcmus.edu.vn

Nội dung

- Giới thiệu, cài đặt
- Biến, hằng, các kiểu dữ liệu cơ bản (int, float, boolean, string)
- Nhập xuất cơ bản
- Cấu trúc dữ liệu cơ bản (list, tuple, set, dictionary)
- Ép kiểu dữ liệu
- Cấu trúc rẽ nhánh
- Cấu trúc lặp
- Hàm
- Xử lý tập tin
- Chú thích, source code encoding
- Scope

Giới thiệu

- Ngôn ngữ lập trình cấp cao phổ biến
  - Data science: AI, Machine Learning, Deep Learning, ...
  - Software: embedded system, back-end (Django), …
- Tính chất tuần tự
- Trình thông dịch (interpreter)
- Phiên bản được hỗ trợ: Python 3

Cài đặt

- Cài đặt:
  - Trình thông dịch Python (Windows, Linux, MacOS)
  - Editor: VSCode, Sublime, Notepad++, Vim, …
  - Thực thi:
    - source code → interpreter → execute
    - shell → interactive interface
- Công cụ hỗ trợ khác:
  - Online: Google Colab
  - Offline: Jupyter Lab

Biến, hằng

- Biến:
  - đối tượng lưu trữ giá trị, thông tin trong chương trình
  - giá trị có thể thay đổi qua phép gán
  - đặt tên viết thường, hoặc theo chuẩn (PEP8)
- Ví dụ:

  age = int(26)

  fullName = ‘Ng Thanh An’

  house_price = 2e9

Biến, hằng

- Hằng:
  - đối tượng lưu trữ giá trị, thông tin trong chương trình
  - giá trị không thay đổi xuyên suốt chương trình
  - đặt tên viết hoa toàn bộ
- Ví dụ:

  PI = 3.14

  API_KEY = ‘5a53bd0086c539c8’

  MIN_YEAR = 1970

Kiểu dữ liệu cơ bản

- int: kiểu dữ liệu số nguyên có dấu (tuổi, số lượng, đại lượng tự nhiên, ...)
- Khai báo:
  - <tên biến> = <giá trị nguyên> age = 26
  - <tên biến> = int(<giá trị>) num = int(1995)

- Thao tác: +, -, *, /, //, %, **
- VD: khai báo biến a = 26, b = 5, in ra type và kết quả tất cả các phép tính kể trên.

Kiểu dữ liệu cơ bản

- float: kiểu dữ liệu số thực (số đo, tính toán số học, đại lượng liên tục,...)
- Khai báo:
  - <tên biến> = <giá trị thực> x = 5. y = 1e5
  - <tên biến> = float(<giá trị>) num = float(3.14)

- Thao tác: +, -, *, /, //, %, **
- VD: khai báo biến a = 26.5, b = 5.7, in ra type và kết quả tất cả các phép tính kể trên.

Kiểu dữ liệu cơ bản

- boolean: kiểu dữ liệu luân lý (tính chất đúng sai, kết quả kiểm định,...)
- Khai báo:
  - <tên biến> = <giá trị True/False> check = True
  - <tên biến> = bool(<giá trị>) valid = bool(0)
- Thao tác: not, and, or
- VD: khai báo biến a = True, b = False, in ra type và kết quả tất cả các thao tác kể trên.
- Lưu ý: các phép so sánh, kiểm tra hay dùng:
  - <, <=, ==, >=, >, !=
  - in, not in

Kiểu dữ liệu cơ bản

- string: chuỗi ký tự (tên, địa chỉ, định danh, văn bản,...)
- Khai báo:
  - <tên biến> = <giá trị> name = ‘Thành An’
  - <tên biến> = str(<giá trị>) phone = str(‘0123456’)

- Thao tác: [], +, *, len, upper, lower, capitalize, replace, strip, find
- VD: khai báo biến h = ‘Nguyễn’, l = ‘thành’, t = ‘An’
  - in ra type, độ dài 3 chuỗi
  - ghép thành tên hoàn chỉnh, có viết hoa

Kiểu dữ liệu cơ bản

- String format: tạo ra chuỗi ký tự từ template và các giá trị
  - ‘<template>’%(<danh sách giá trị>)
  - ‘<template>’.format(<danh sách giá trị>)
- VD:
  - ‘%s %d’%(‘Thành An’, 26)
- placeholders: %d, %f, %s, %g, …
- Lưu ý:
  - ‘’ và “”
  - ký tự đặc biệt: \n, \t, \\, %%

Nhập xuất cơ bản

- Nhập:
  - input(<string>) → string
  - VD: name = input(‘your name:’)

    age = int(input(‘age: ’))
- Xuất
  - print(<value>, <value>, ...)
  - VD: print(‘xin chào’)

    print(‘kết quả: ’, process(data))

Cấu trúc dữ liệu cơ bản - list

l = [1, 2, 3]

- Mảng các phần tử với kiểu dữ liệu bất kỳ.
- Truy cập phần tử (đọc/gán) bằng chỉ số, đếm từ 0. VD: l[0], l[-1]
- Thao tác:
  - append, insert
  - remove, pop
  - len
  - sort, reverse
  - index, count
  - split theo chỉ số. VD: l[0:2]

Cấu trúc dữ liệu cơ bản - tuple

t = (‘an’, 26)

- Bộ các phần tử với ý nghĩa cụ thể.
- Không gán/thêm/xoá phần tử. Chỉ đọc giá trị. VD: t[0], t[1]
- Thao tác:
  - len
  - index, count
  - gộp bộ:

u = (‘nguyễn’, ‘thành’) + t

Cấu trúc dữ liệu cơ bản - set

s = {1, 2, 5}

- Tập hợp các phần tử duy nhất.
- Không truy cập bằng chỉ số.
- Thao tác:
  - len
  - add, remove
  - intersection, union, difference

Cấu trúc dữ liệu cơ bản - dictionary

d = {‘name’:’an’, ‘age’:26, ‘job’:’IT’}

- Tập hợp các phần tử theo dạng <key>:<value>. Key là duy nhất.
- Truy cập giá trị bằng key. VD: d[‘name’] = ‘thành an’
- Thao tác:
  - len
  - thêm giá trị: d[<key>] = <value>
  - pop
  - keys, values

Ép kiểu dữ liệu

- Thao tác chuyển đổi dữ liệu từ kiểu/cấu trúc này sang kiểu/cấu trúc khác.
  - int ↔ float int, float, bool ↔ string
  - list ↔ tuple ↔ set
- Ví dụ:
  - gpa = float(‘9.0’)
  - values = [1, 1, 2, 3, 3]
  - unique_values = set(values) # {1, 2, 3}
- Không phải kiểu/cấu trúc nào cũng chuyển đổi qua lại được cho nhau.
- Dữ liệu có thể bị thay đổi khi ép kiểu.

Cấu trúc rẽ nhánh

- Các keyword: if elif else
- Cú pháp tổng quát:

```
if <mệnh đề>:
    <khối lệnh xử lý>
elif <mệnh đề>:
    <khối lệnh xử lý>
else:
    <khối lệnh xử lý>
pass
```

Cấu trúc rẽ nhánh

- Ví dụ:

```
if  temp > 30:
        print(‘it is so hot’)
elif temp > 20:
        print(‘it is cool’)
else:
        print(‘it is cold’)
pass
```

Cấu trúc lặp - while

- Cấu trúc tổng quát

```
while <mệnh đề điều kiện>:
    <khối lệnh xử lý>
pass
```

- Ví dụ:

```
i = 10
while i > 0:
    print(i)
    i = i - 1
```

Cấu trúc lặp - for

- Cấu trúc tổng quát:

```
for <biến tạm> in <container>:
    <khối lệnh xử lý>
pass
```

- Ví dụ:

```
for name in [‘an’, ‘binh’, ‘chi’]:
    print(‘xin chao ‘ + name)
```

Cấu trúc lặp - lưu ý

- Lệnh thường dùng:
  - break: ngắt lặp
  - continue: qua lần lặp tiếp theo
- for

```
a = [1, 2, 3, 4, 5]

b = [x**2 for x in a if x%2==0]

d = {str(x):x**2 for x in a}

for (i, item) in enumerate(a):

    …
```

Hàm

- Cấu trúc khai báo:
  - Thủ tục

```
def <tên hàm>(<danh sách tham số>):
    <khối lệnh xử lý>
    pass
```

  - Hàm

```
def <tên hàm>(<danh sách tham số>):
    <khối lệnh xử lý>
    return <giá trị kết quả>
```

Hàm

- Ví dụ:

```
def add(a, b):
    return a + b


def getDistance(A, B):
    x1, y1 = A
    x2, y2 = B
    return math.sqrt((x1-x2)**2 + (y1-y2)**2)
```

Hàm - tham số mặc định

- Cú pháp:

```
def <tên hàm>(<TS 1>,…, <TS m>=<giá trị>, <TS n>=<giá trị>, ...):
    <khối lệnh xử lý>
    pass

def <tên hàm>(<TS 1>,…, <TS m>=<giá trị>, <TS n>=<giá trị>, ...):
    <khối lệnh xử lý>
    return <giá trị>
```

Hàm - tham số mặc định

- Ví dụ:

```
def add(a, b, c=0):
    return a + b + c
def readDataFile(filename=’data.txt'):
    …
    return 1
def solveQuadraticEquation(a, b, c=0):
    …
    return …
```

Hàm - tham số mặc định

- Trường hợp 1: Truyền đầy đủ tham số của hàm **theo thứ tự**

```
solveQuadraticEquation(1, -2, 1)
```

- Trường hợp 2: Chỉ truyền giá trị cho các tham số không có giá trị mặc định

```
solveQuadraticEquation(1, -2)          # c=0
```

- Trường hợp 3: Truyền giá trị cho đúng các biến bằng cách đặc tả tên biến (không cần thứ tự)

```
solveQuadraticEquation(c=1, b=-2, a=0)
```

Xử lý tập tin

- Quy trình thao tác: mở → đọc/ghi → đóng.
- Đọc dữ liệu: open(<filename>, mode='rt')
- Ghi dữ liệu: open(<filename>, mode='wt')
- Thêm dữ liệu: open(<filename>, mode='at')
- Trường hợp lỗi:
  - Tập tin không tồn tại?
  - Ghi vào tập tin đã tồn tại?

Chú thích

- Là những ghi chú trong mã nguồn, không được thông dịch/biên dịch, diễn giải nội dung/thuật toán.
- Có hai loại chú thích
  - single line: #
  - multiple lines: ‘’’ ‘’’

```
# declare variables
‘’’ TODO 01:

   declare two variables...

‘’’
```

Source code encoding

- Đôi khi trong mã nguồn có ký tự đặc biệt (unicode) thì cần khai báo encoding cho tập tin để tránh xảy ra lỗi khi thông dịch hoặc ảnh hướng tới các trình biên tập.
- Cú pháp: # -*- coding: <encoding name> -*-
- Ví dụ:

```
# -*- coding: latin-1 -*-

# -*- coding: iso-8859-15 -*-

# -*- coding: ascii -*-
```

Scope

- Scope được hiểu là phạm vi khả dụng của các thực thể (biến, hằng, hàm, ...) trong chương trình.
- Ví dụ:
  - scope toàn cục
  - scope của một khối lệnh
    - scope của một khối hàm
    - scope của một khối lặp
    - scope của một khối điều kiện

Scope

- Ví dụ: scope của khối hàm

```
def doSth(b):
    a = 1
    return a + b


print(doSth(10))
print(a)
```

```
def doSth(b):
    a = 1
    def doSth2(x):
        return a + x
    return a + doSth2(b)


print(doSth(5))
print(a, b)
```