Lab 01 - Python

Linh Truong

Table of contents

1 Python

Python là một ngôn ngữ lập trình phổ biến hiện nay, được tạo ra bởi Guido van Rossum và phát hành vào năm 1991. Python là ngôn ngữ thường được dùng cho các lĩnh vực như: phát

triển web, viết phần mềm, tính toán, viết kịch bản hệ thống, phân tích dữ liệu...

Tài liệu này hướng dẫn các bạn tiếp cận Python ở góc độ người đã từng học qua một ngôn ngữ lập trình, vì vậy các nội dung sẽ chủ yếu trình bày dưới dạng các ví dụ. Môi trường để thực hành là file notebook (`.ipynb`), các bạn hoàn toàn có thể chạy, ghi chú và chỉnh sửa trên file.

Tài liệu gồm những nội dung sau:

- Basic Data types (Kiểu dữ liệu cơ bản)
  - Numbers (kiểu số)
  - Boolean
  - String (kiểu chuỗi)
  - Input, Output (Nhập, xuất)
  - List (kiểu danh sách)
- Control flow (Luồng điều khiển)
  - if
  - while
  - for
  - function (hàm)
- Containers
- Built-in functions (một số hàm có sẵn thông dụng)
- Tham khảo

1.1 BASIC DATA TYPES

Numbers

Số nguyên và số thực trong Python hoạt động như trong các ngôn ngữ khác.

```
1 + 1
```

2

```
50 - 5 * 6
```

20

```
17 / 3  # classic division returns a float
```

```
5.666666666666667
```

```
17 // 3  # floor division discards the fractional part
```

```
5
```

```
5 ** 2  # 5 squared
```

```
25
```

```
4 % 2
```

```
0
```

```
5 % 2
```

```
1
```

```
(2 + 3) * (5 + 5)
```

```
50
```

```
# Sử dụng toán tử '=' để gán giá trị cho biến, không cần phải khai báo biến trước khi sử d
width = 20
height = 5 * 9
width * height
```

```
900
```

```
# sử dụng biến mà không gán giá trị thì sẽ gặp thông báo lỗi
n
```

```
NameError: name 'n' is not defined
```

Boolean

```
t = True
f = False
print(type(t))
print(t and f)
print(t or f)
print(not f)
print(t != f)    # XOR
```

```
<class 'bool'>
False
True
True
True
```

```
x = 3
print(x%2 != 0)
print(x>0 and x%5==0)
print(x==3 or x==9)
```

```
True
False
True
```

String

```
hello = 'hello'               # String literals can use single quotes
world = "world"               # or double quotes; it does not matter.
print(hello)                  # Prints "hello"
print(len(hello))             # String length; prints "5"
```

```
hello
5
```

```
hw = hello + ' ' + world      # String concatenation
print(hw)                     # Prints "hello world"
```

```
hello world
```

```
hw12 = '%s %s %d' % (hello, world, 12)      # sprintf style string formatting
print(hw12)                                 # prints "hello world 12"
hw12 = "{} {} {}".format(hello, world, 12)  # sprintf style string formatting
print(hw12)                                 # prints "hello world 12"
```

```
hello world 12
hello world 12
```

Xâu kí tự trong Python là một lớp đối tượng và có rất nhiều phương thức hữu ích; ví dụ:

```
s = "hello"
print(s.capitalize())  # Capitalize a string; prints "Hello"
print(s.upper())       # Convert a string to uppercase; prints "HELLO"
print(s.rjust(7))      # Right-justify a string, padding with spaces; prints "  hello"
print(s.center(7))     # Center a string, padding with spaces; prints " hello "
print(s.replace('l', '(ell)'))  # Replace all instances of one substring with another;
                                # prints "he(ell)(ell)o"
print('  world '.strip())  # Strip leading and trailing whitespace; prints "world"
```

```
Hello
HELLO
  hello
 hello
he(ell)(ell)o
world
```

Bạn có thể tìm hiểu thêm vì các phương thức của lớp xâu kí tự tại đây

List

```
[1,2,3] # Elements are the same data type
```

```
[1, 2, 3]
```

```
['hi',1,[1,2]]  # Elements are the different data types
```

```
['hi', 1, [1, 2]]
```

```
nest = [1,2,3,[4,5,['target']]]

print(nest[3])
print(nest[3][2])
print(nest[3][2][0])
```

```
[4, 5, ['target']]
['target']
target
```

```
my_list = ['a','b','c']   # declare a new variable
```

```
my_list.append('d')     # append an element
my_list
```

```
['a', 'b', 'c', 'd']
```

```
# Access single element by id
print(my_list[0])
print(my_list[2])
```

```
a
c
```

```
# Access a subset of list using slicing (':' operator)
print(my_list[1:])
print(my_list[1:3])
```

```
['b', 'c', 'd']
['b', 'c']
```

```
# Change value
my_list[0] = 'NEW'
my_list
```

```
['NEW', 'b', 'c', 'd']
```

```
# Kiểm tra phần tử 'NEW' có thuộc danh sách
'NEW' in my_list
```

```
True
```

```
# Xóa phần tử ở vị trí thứ 2
del my_list[2]
print(my_list)
```

```
['NEW', 'b', 'd']
```

```
# Print each element in list
for x in my_list:
  print(x)
```

```
NEW
b
d
```

Bạn có thể tìm hiểu thêm vì các phương thức của list tại đây

Input, Output

Output

```
x = 'hello'
```

```
x
```

```
'hello'
```

```
print(x)
```

```
hello
```

```
num = 12
name = 'Sam'

print(name, num)
print(name, num, sep=',') # Adding the seperator ',' between 2 variables name and num
print(name, ' has the number ', num)
```

```
Sam 12
Sam,12
Sam  has the number  12
```

```
print(f'My number is: {num}, and my name is: {name}')
```

```
My number is: 12, and my name is: Sam
```

```
print('My number is: {}, and my name is: {}'.format(num,name))
```

```
My number is: 12, and my name is: Sam
```

```
print('My number is: %d, and my name is: %s' % (num,name))
```

```
My number is: 12, and my name is: Sam
```

Input

```
# Nhập một chuỗi
print('Enter your name:')
x = input()
print('Hello, ' + x)
print(type(x))
```

```
Enter your name:

 Peter Parker

Hello, Peter Parker
<class 'str'>
```

```
# Nhập một số nguyên
age = int(input('Enter your age: '))
print('Your age is', age)
print(type(age))
```

```
Enter your age:  19

Your age is 19
<class 'int'>
```

```
# Nhập một số thực
weight = float(input('Enter your weight: '))
print('Your weight is', weight)
print(type(weight))
```

```
Enter your weight:  68.7

Your weight is 68.7
<class 'float'>
```

```
# Nhập một danh sách với số lượng xác định số phần tử số nguyên
# creating an empty list
lst = []

n = int(input("Enter number of elements : "))

for i in range(0, n):
    ele = int(input())
    # adding the element
    lst.append(ele)

print(lst)
```

```
Enter number of elements :  4
 2
 78
 -4
 9

[2, 78, -4, 9]
```

```
# Nhập một list các phần tử số thực cách nhau bởi dấu ,
input_string = '1.5, -8.25, 9, 2.08'

lst1 = [float(x) for x in input_string.split(',')]
print(lst1)
```

```
[1.5, -8.25, 9.0, 2.08]
```

```
# Nhập một list các phần tử số thực cách nhau bởi dấu , (cách khác)
input_string = '1.5, -8.25, 9, 2.08'

lst2 = list(map(float, input_string.split(',')))
print(lst2)
```

```
[1.5, -8.25, 9.0, 2.08]
```

1.2 CONTROL FLOW

if, if-else, if-elif-else

Cú pháp câu lệnh if:

```
if logical_condition:
    statement(s)
```

Cần phải đặt dấu : ở cuối câu lệnh điều kiện, và các câu lệnh con ở các dòng tiếp theo phải thụt vào trong so với câu lệnh if (thông thường là 4 khoảng trắng)

```
# if
a = 2000
b = 1999

if a > b:
  print('a is greater than b')
```

```
a is greater than b
```

```
# if-else
a = 2000
b = 1999

if a < b:
  print('b is greater than a')
else:
  print('a is greater than b')
```

```
a is greater than b
```

```
# if-elif-else
a = 2000
b = 2000

if b > a:
  print('b is greater than a')
elif a == b:
  print('a and b are equal')
else:
  print('a is greater than b')
```

```
a and b are equal
```

```
# Nested if (câu lệnh điều kiện lồng nhau)
a = 1999
b = 2000

if a > b:
    print('a > b')
elif a < b:
    print('a < b')
    if a == 1999:
        print('a = 1999')
    else:
        print('a is not equal to 1999')
else:
    print('a = b')
```

```
a < b
a = 1999
```

while

```
while logical_condition:
    statement(s)
```

```
i = 1
while i < 10:
    print(i*2)
    i = i + 1
print('Mission accomplished!')
```

```
2
4
6
8
10
12
14
16
18
Mission accomplished!
```

for

Với mỗi phần tử của biến **sequence**, câu lệnh statement(s) được thực thi

```
for variable in sequence:
    statement(s)
```

```
for i in range(10):
    print(i)
```

```
0
1
2
3
```

```
4
5
6
7
8
9
```

```
for i in [1, 5, 10, 15]:
    print(i*5)
```

```
5
25
50
75
```

break

Câu lệnh `break` dùng để thoát khỏi vòng lặp

```
for i in range(10):
    print(i)
    if i >= 4:
        break
```

```
0
1
2
3
4
```

continue

Câu lệnh `continue` khiến cho vòng lặp kế tiếp được thực hiện, ngay cả khi vòng lặp hiện tại chưa được thực hiện xong.

```
for i in range(10):
    if i == 2:
        print('Ignoring 2')
        continue
```

```
    else:
        print(i)
```

```
0
1
Ignoring 2
3
4
5
6
7
8
9
```

Function

Cú pháp định nghĩa hàm:

```
def function_name( parameters... ):
   "function_docstring"
   function_suite...
   ...
   return [return_value]
```

Trong đó:

- function_name: tên hàm
- parameters: danh sách các tham số
- function_docstring: ghi chú, giải thích ý nghĩa của hàm. Nên có để viết code cho dễ đọc
- function_suite: chứa các câu lệnh sẽ thực hiện trong hàm
- return_value: giá trị trả về của hàm nếu có

```
# Define a function
def my_function():
  "This is a testing function"
  print("Hello from a function")
```

```
# Call the defined function
my_function()
```

```
Hello from a function
```

```
# Define and call a function with a parameter
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emily")
my_function("Tobias")
my_function("Linus")
```

```
Emily Refsnes
Tobias Refsnes
Linus Refsnes
```

```
# Define and call a function with two parameters
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

```
Emil Refsnes
```

```
# Passing a list as an argument
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

```
apple
banana
cherry
```

```
# Function with return value
def my_function(x):
  return 5 * x
```

```
print(my_function(3))
print(my_function(5))
print(my_function(9))
```

```
15
25
45
```

Ta có thể định nghĩa hàm nhận tham số tùy chọn, như sau:

```
# Define a function with default parameter 'loud'
def hello(name, loud=False):
    if loud:
        print('HELLO, %s!' % name.upper())
    else:
        print('Hello, %s' % name)

hello('Bob') # Prints "Hello, Bob"
hello('Fred', loud=True)  # Prints "HELLO, FRED!"
```

```
Hello, Bob
HELLO, FRED!
```

1.3 BUILT-IN FUNCTIONS

```
# Hàm lấy độ dài
mylist = ["apple", "banana", "cherry"]
n = len(mylist)
print(n)
```

```
3
```

```
# Hàm xác định kiểu dữ liệu của biến
a = ('apple', 'banana', 'cherry')
b = "Hello World"
c = 33
```

```
print(f'var: {a}, type:{type(a)}')
print(f'var: {b}, type:{type(b)}')
print(f'var: {c}, type:{type(c)}')
```

```
var: ('apple', 'banana', 'cherry'), type:<class 'tuple'>
var: Hello World, type:<class 'str'>
var: 33, type:<class 'int'>
```

```
# Hàm lũy thừa
pow(2, 3)
```

```
8
```

`round()` sử dụng để làm tròn phần thập phân

```
# Hàm làm tròn
print(round(3.14159265359, 0))
print(round(3.14159265359, 2))
```

```
3.0
3.14
```

```
# Tính giá trị tuyệt đối
abs(-3.4)
```

```
3.4
```

Bạn có thể tìm hiểu thêm chi tiết các hàm có sẵn trong python tại đây

1.4 CONTAINERS

Ngoài các kiểu dữ liệu cơ bản trong Python còn một số kiểu dữ liệu dùng để chứa nhiều phần tử được gọi là các containers (lớp chứa). Các container có sẵn trong python là có sẵn là: `list`, `dict`, `set` và `tuple`.

list

Trong Python, `list` tương tự như khái niệm mảng trong các ngôn ngữ lập trình khác, nhưng `list` trong Python có thể thay đổi kích thước và chứa các phần tử có kiểu dữ liệu khác nhau.

```
xs = [3, 1, 2]    # Create a list
print(xs, xs[2])  # Prints "[3, 1, 2] 2"
print(xs[-1])     # Negative indices count from the end of the list; prints "2"
xs[2] = 'foo'     # Lists can contain elements of different types
print(xs)         # Prints "[3, 1, 'foo']"
xs.append('bar')  # Add a new element to the end of the list
print(xs)         # Prints "[3, 1, 'foo', 'bar']"
x = xs.pop()      # Remove and return the last element of the list
print(x, xs)      # Prints "bar [3, 1, 'foo']"
```

```
[3, 1, 2] 2
2
[3, 1, 'foo']
[3, 1, 'foo', 'bar']
bar [3, 1, 'foo']
```

Bạn có thể tìm hiểu thêm chi tiết tại đây

Slicing

Ngoài việc truy cập từng phần tử của `list` , Python còn cung cấp cú pháp để truy cập các `list` con như dưới đây được gọi là `slicing`.

```
nums = list(range(5))     # range is a built-in function that creates a list of integers
print(nums)               # Prints "[0, 1, 2, 3, 4]"
print(nums[2:4])          # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print(nums[2:])           # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print(nums[:2])           # Get a slice from the start to index 2 (exclusive); prints "[0,
print(nums[:])            # Get a slice of the whole list; prints "[0, 1, 2, 3, 4]"
print(nums[:-1])          # Slice indices can be negative; prints "[0, 1, 2, 3]"
nums[2:4] = [8, 9]        # Assign a new sublist to a slice
print(nums)               # Prints "[0, 1, 8, 9, 4]"
```

```
[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
```

```
[0, 1]
[0, 1, 2, 3, 4]
[0, 1, 2, 3]
[0, 1, 8, 9, 4]
```

Slicing sẽ được đề cập lại ở phần sau, sử dụng cùng với các mảng numpy.

Duyệt list

Bạn có thể duyệt qua các phần tử của `list` như sau:

C1: Dùng từ khóa `in`

```
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)   # Prints [0, 1, 4, 9, 16]
```

```
[0, 1, 4, 9, 16]
```

C2: Sử dụng các kết quả của hàm `range(...)`

```
nums = [0, 1, 2, 3, 4]
squares = []
for i in range(len(nums)):
    squares.append(nums[i]**2)
print(squares)
```

```
[0, 1, 4, 9, 16]
```

```
nums = [0, 1, 2, 3, 4]
squares = []
for i in range(2, len(nums)):
    squares.append(nums[i]**2)
print(squares)
```

```
[4, 9, 16]
```

C3: sử dụng giống cú pháp Select..From..Where như trong SQL

```
nums = [0, 1, 2, 3, 4]
squares = [x ** 2 for x in nums]
print(squares)   # Prints [0, 1, 4, 9, 16]
```

```
[0, 1, 4, 9, 16]
```

Các list comprehension cũng có thể chứa các điều kiện:

```
nums = [0, 1, 2, 3, 4]
even_squares = [x ** 2 for x in nums if x % 2 == 0]
print(even_squares)  # Prints "[0, 4, 16]"
```

```
[0, 4, 16]
```

C4: Trong trường hợp bạn muốn truy cập đến chỉ mục của phần tử trong thân vòng lặp, sử dụng hàm dựng sẵn `enumerate`:

```
animals = ['cat', 'dog', 'monkey']
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
```

```
#1: cat
#2: dog
#3: monkey
```

dict

Một từ điển `dict` chứa cặp (key, value), tương tự như `Map` trong ngôn ngữ `Java`, trong đó các `key` là duy nhất. Bạn có thể sử dụng nó như sau:

```
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data
print(d['cat'])       # Get an entry from a dictionary; prints "cute"
print('cat' in d)     # Check if a dictionary has a given key; prints "True"
d['fish'] = 'wet'     # Set an entry in a dictionary
print(d['fish'])      # Prints "wet"
# print(d['monkey'])  # KeyError: 'monkey' not a key of d
print(d.get('monkey', 'N/A'))  # Get an element with a default; prints "N/A"
print(d.get('fish', 'N/A'))    # Get an element with a default; prints "wet"
```

```
del d['fish']         # Remove an element from a dictionary
print(d.get('fish', 'N/A')) # "fish" is no longer a key; prints "N/A"
```

```
cute
True
wet
N/A
wet
N/A
```

Tìm hiểu thêm về `dict` tại đây.

Duyệt dict

Có thể dễ dàng duyệt qua các khóa trong một từ điển:

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    legs = d[animal]
    print('A %s has %d legs' % (animal, legs))
```

```
A person has 2 legs
A cat has 4 legs
A spider has 8 legs
```

Nếu bạn muốn truy cập đến các khóa và các giá trị tương ứng, sử dụng phương thức `items`:

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
```

```
A person has 2 legs
A cat has 4 legs
A spider has 8 legs
```

Tạo dict mới từ dict có sẵn thỏa mãn một số điều kiện. Ví dụ:

```
nums = [0, 1, 2, 3, 4]
even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}
print(even_num_to_square)  # Prints "{0: 0, 2: 4, 4: 16}"
```

```
{0: 0, 2: 4, 4: 16}
```

set

Một `set` là một tập hợp không xét thứ tự của các phần tử riêng biệt. Xem xét một ví dụ đơn giản sau:

```
animals = {'cat', 'dog'}
print('cat' in animals)   # Check if an element is in a set; prints "True"
print('fish' in animals)  # prints "False"
animals.add('fish')       # Add an element to a set
print('fish' in animals)  # Prints "True"
print(len(animals))       # Number of elements in a set; prints "3"
animals.add('cat')        # Adding an element that is already in the set does nothing
print(len(animals))       # Prints "3"
animals.remove('cat')     # Remove an element from a set
print(len(animals))       # Prints "2"
```

```
True
False
True
3
3
2
```

Bạn có thể tìm hiểu thêm về `set` tại đây.

Duyệt qua `set` tương tự như duyệt qua `list`; tuy nhiên vì `set` không có thứ tự, bạn không thể truy cập các phần tử thông qua chỉ mục (`id`)

```
animals = {'cat', 'dog', 'fish'}
print(animals)
```

```
{'cat', 'fish', 'dog'}
```

```
# This statement will raise error
animals[1]
```

```
TypeError: 'set' object is not subscriptable
```

```
# Duyệt bằng toán tử in
for a in animals:
  print(a)
```

```
cat
fish
dog
```

```
# Duyệt sử dụng hàm `enumerate()`
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
```

```
#1: cat
#2: fish
#3: dog
```

```
# Cách khác
from math import sqrt

nums = {int(sqrt(x)) for x in range(30)}
print(nums)  # Prints "{0, 1, 2, 3, 4, 5}"
```

```
{0, 1, 2, 3, 4, 5}
```

tuple

Một tuple là một danh sách có thứ tự của các phần tử và danh sách này không thay đổi được. Một `tuple` có nhiều điểm tương tự như `list`; điểm khác biệt quan trọng nhất là `tuple` có thể được dùng như các khóa của `dict` hay các phần tử của `set`, `list` thì không thể. Sau đây là một ví dụ:

```python
d = {(x, x + 1): x for x in range(10)}  # Create a dictionary with tuple keys
t = (5, 6)        # Create a tuple
print(type(t))    # Prints "<class 'tuple'>"
print(d[t])       # Prints "5"
print(d[(1, 2)])  # Prints "1"
```

```
<class 'tuple'>
5
1
```

Tài liệu chi tiết về `tuple` có thể xem tại đây.

1.5 TÓM TẮT

- Kiểu dữ liệu cơ bản

Type	Description
int	Integer
float	Floating-point number
str	String
bool	Boolean (True or False)

- Các toán tử

Toán tử	Ghi chú
+	cộng
-	trừ
/	chia
//	chia lấy phần nguyên
*	nhân
%	chia lấy phần dư
**	lũy thừa

- Luồng điều khiển
  - Câu lệnh lựa chọn (`if`, `if-else`, `if-elif-else`)

  - Câu lệnh lặp (`while, for`)
  - Hàm

- Các lớp chứa (containers)

  - `list`
  - `dict`
  - `set`
  - `tuple`

- Các hàm thông dụng (built-in funciton)

Hàm	Ghi chú
len()	lấy độ dài
type()	xem kiểu dữ liệu
range()	trả về dãy số có thứ tự
str()	ép kiểu sang kiểu chuỗi
pow()	lũy thừa
round()	làm tròn
abs()	tính trị tuyệt đối

1.6 THAM KHẢO

Tài liệu sẽ hướng dẫn các chủ đề cần thiết để phục vụ môn học, nếu có nhu cầu tìm hiểu và học thêm các bạn có thể tham khảo từ các nguồn sau:

- The Python Tutorial (official)
- W3School Python
- Tutorialspoint

1.7 CHANGELOG

- 2023-09-05: first release