Giới thiệu về ngôn ngữ lập trình Python 1

Hoàng Nam Dũng

Khoa Toán - Cơ - Tin học, Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội

Cài đặt

Cách 1 (dùng offline) - Cài Anaconda

- Vào https://www.anaconda.com/download/, download phiên bản Python 3.7
- Tiến hành cài đặt như hướng dẫn tại https://mechasolution.vn/Blog/bai-2-cai-dat-anaconda
- Hướng dẫn sử dụng Anaconda: http://docs.anaconda.com/anaconda/user-guide/getting-started/

Cách 2 (dùng online) - Dùng colab
https://colab.research.google.com

- Ưu điểm: không cần tự cài các gói cơ bản
- Nhược điểm: phải cần có mạng để chạy.

(Một số) sách về Python

Có thể tìm tại http://libgen.is/

- Eric Matthes, *Python Crash Course: A Hands-On, Project-Based Introduction to Programming*, 2015
- Dan Bader, *Python Tricks: A Buffet of Awesome Python Features*, 2017
- Luciano Ramalho, *Fluent Python: Clear, Concise, and Effective Programming*, O'Reilly, 2015

Danh sách nhiều tài liệu tham khảo ở nhiều cấp độ khác nhau: https://docs.python-guide.org/intro/learning/

Tài liệu tham khảo (online)

Ngôn ngữ lập trình Python:
https://www.programiz.com/python-programming

Các khóa học online

- Python tại MIT (có video) https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programmin
- Python tại Stanford (có video) https://stanfordpython.com/

Tài liệu tham khảo (online)

Python cho tính toán khoa học:

- https://www.scipy-lectures.org/index.html
- Hans Fangohr, *Introduction to Python for Computational Science and Engineering*, 2016 – link download: https://www.southampton.ac.uk/~fangohr/teaching/python/book.html (có code)

Các tài liệu khác

- https://www.learnpython.org/
- 90% python trong 90 phút https://www.slideshare.net/MattHarrison4/learn-90
- Google

Tài liệu tiếng Việt[1]

- https://toidicode.com/python-co-ban
- Video https://www.youtube.com/playlist?list=PL33lvabfss1xczCv2BA0SaNJHu_VXsFtg

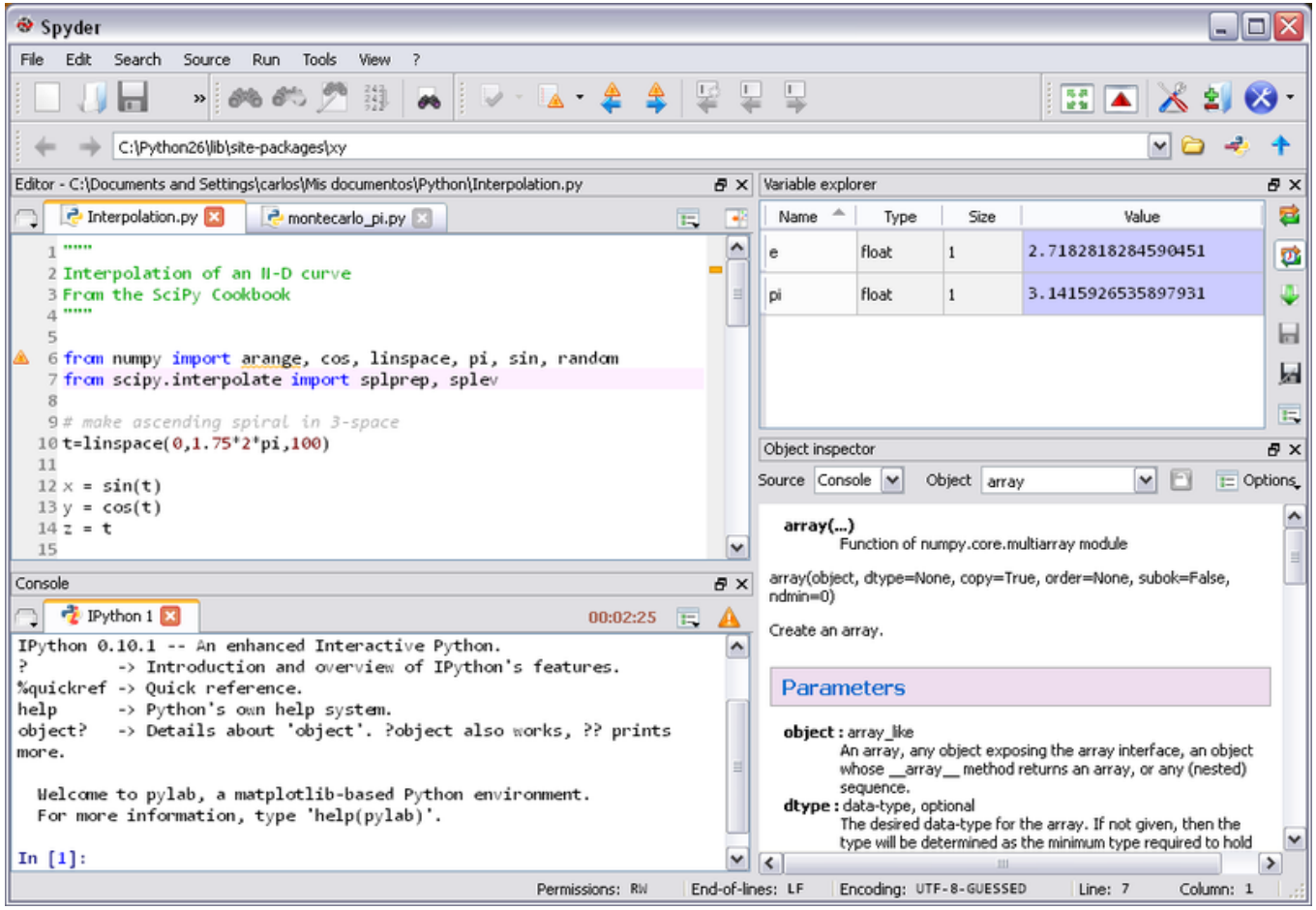
[1] Không khuyến khích sử dụng

Bài tập - tự luyện tập

https://www.hackerrank.com/domains/python

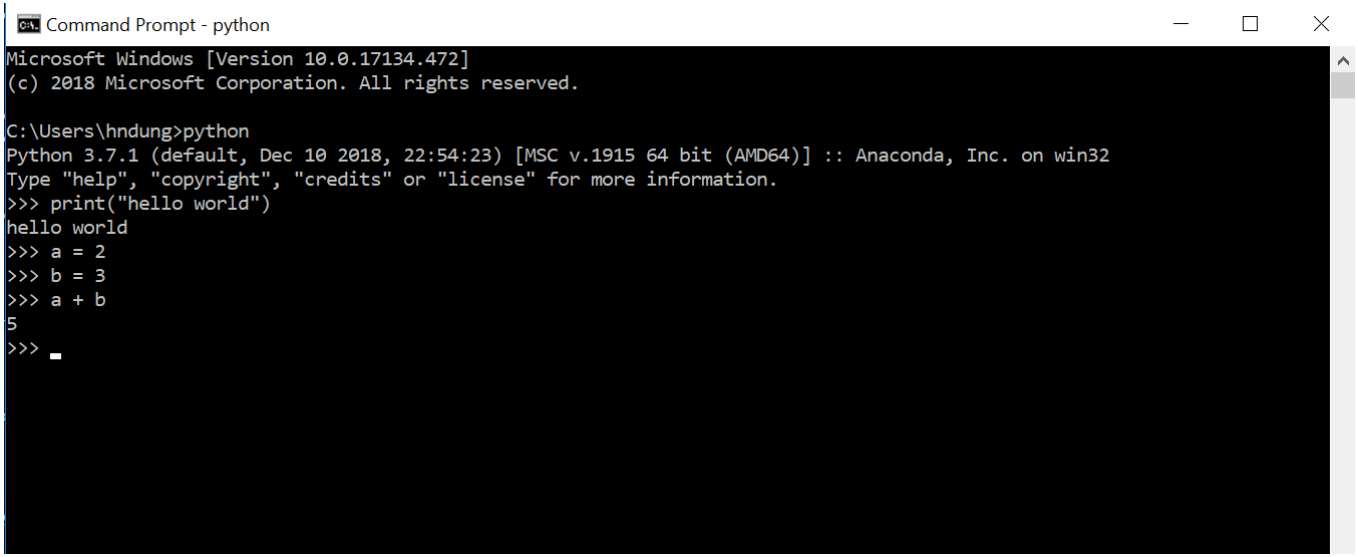
Cách chạy code python

- Dùng colab: Xem trên lớp
- Dùng IDE (môi trường phát triển tích hợp) Spyder.



Cách chạy code python

- Trên Windows mở cmd (command promt), đánh lệnh python và lập trình trực tiếp qua dòng lệnh.



- Sử dụng bất kì phần mềm soạn thảo nào, lập trình và lưu file dưới dạng *filename.py*. Trên Windows mở cmd (command promt) và đánh lệnh python *filename.py*.

Chương trình đầu tiên

```
>>> # Add two numbers
>>> num1 = 3
>>> num2 = 5
>>> sum = num1+num2
>>> print(sum)
8
```

Chương trình đầu tiên

```
>>> # Add two numbers
>>> num1 = 3
>>> num2 = 5
>>> sum = num1+num2
>>> print(sum)
8

>>> # Hello world
>>> print("hello world")
hello world
```

Chương trình đầu tiên

```
>>> # Add two numbers
>>> num1 = 3
>>> num2 = 5
>>> sum = num1+num2
>>> print(sum)
8

>>> # Hello world
>>> print("hello world")
hello world

>>> # Hello name
>>> name = "Tuấn"
>>> print("hello {}".format(name))
hello Tuấn
```

Biến

https://www.programiz.com/python-programming/variables-constants-literals

Các kiểu dữ liệu - Số (number)

```
>>> a = 5
>>> print(a, "is of type", type(a))
5 is of type <class 'int'>

>>> a = 2.0
>>> print(a, "is of type", type(a))
2.0 is of type <class 'float'>

>>> a = 1+2j
>>> print(a, "is complex number?", isinstance(1+2j,complex))
(1+2j) is complex number? True
```

List - Khởi tạo

```
>>> # empty list
>>> my_list = []
>>> # list of integers
>>> my_list = [1, 2, 3]
>>> # list with mixed datatypes
>>> my_list = [1, "Hello", 3.4]
>>> # nested list - list chứa thành phần là các list
>>> my_list = ["mouse", [8, 4, 6], ['a']]
```

List - Truy cập

List index

```
>>> my_list = ['p','r','o','b','e']
>>> print(my_list[0])
p
>>> print(my_list[2])
o
>>> print(my_list[4])
e
>>> # Chỉ số âm: phần tử cuối cùng
>>> print(my_list[-1])
e
>>> # phần tử thứ 5 từ cuối lên
>>> print(my_list[-5])
p
```

List - Slicing

```
>>> my_list = ['p','r','o','g','r','a','m','i','z']
>>> # elements 3rd to 5th
>>> print(my_list[2:5])
['o', 'g', 'r']
>>> # elements beginning to 4th
>>> print(my_list[:-5])
['p', 'r', 'o', 'g']
>>> # elements 6th to end
>>> print(my_list[5:])
['a', 'm', 'i', 'z']
>>> # elements beginning to end
>>> print(my_list[:])
['p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z']
```

List - Thêm/thay đổi

```
>>> odd = [2, 6, 8]; odd[0] = 1
>>> print(odd)
[1, 6, 8]
>>> odd[1:3] = [5, 7]  # change 2nd to 3th items
>>> print(odd)
[1, 5, 7]
>>> odd.append(9)
>>> print(odd)
[1, 5, 7, 9]
>>> odd.insert(1,3)
>>> print(odd)
[1, 3, 5, 7, 9]
>>> odd.extend([11, 13])
>>> print(odd)
[1, 3, 5, 7, 9, 11, 13]
```

List - concatenation

```
>>> odd = [1, 3, 5]
>>> print(odd + [9, 7, 5])
[1, 3, 5, 9, 7, 5]
>>> print([1, 2, 3] * 3)
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

List - Bớt/xóa

```
>>> my_list = ['p','r','o','b','l','e','m']
>>> # delete one item
>>> del my_list[2]
>>> print(my_list)
['p', 'r', 'b', 'l', 'e', 'm']
>>> # delete multiple items
>>> del my_list[1:5]
>>> print(my_list)
['p', 'm']
>>> # delete entire list
>>> del my_list
```

List - Bớt/xóa

```
>>> my_list = ['p','r','o','b','l','e','m']
>>> my_list.remove('p')
>>> print(my_list)
['r', 'o', 'b', 'l', 'e', 'm']
>>> print(my_list.pop(1))
o
>>> print(my_list)
['r', 'b', 'l', 'e', 'm']
>>> print(my_list.pop())
m
>>> print(my_list)
['r', 'b', 'l', 'e']
>>> my_list.clear()
>>> print(my_list)
[]
```

List

Xem thêm tại

https://www.programiz.com/python-programming/list

Mutable vs. immutable

Simple put: a mutable object can be changed after it is created, and an immutable object can't.

- **Mutable objects**: list, dict, set
- **Immutable objects**: int, float, bool, string, tuple, unicode

Xem medium.com/@meghamohan/mutable-and-immutable-side-of-python-c2145cf72747

Dành cho ai muốn đọc thêm:
`https://inventwithpython.com/blog/2018/02/05/python-tuples-are-immutable-except-when-theyre-mutable/`

Mutable vs. immutable

Hàm id(object) trả về id (identity) của object[2].

Kiểu int là immutable, khi ta thay đổi biến int thì nó sẽ có địa chỉ (id) khác.

```
>>> a = 1
>>> # id - mutable & immutable
>>> print('a = ', a, 'id =', id(a))
a =  1 id = 140734095733584
>>> a = a + 1
>>> print('a = ', a, 'id =', id(a))
a =  2 id = 140734095733616
```

[2] www.programiz.com/python-programming/methods/built-in/id

Mutable vs. immutable

Kiểu list là mutable. Khi thay đổi list, id của nó không đổi.

```
>>> b = []
>>> print(b, id(b))
[] 2882898544968
>>> b.append(2)
>>> print(b, id(b))
[2] 2882898544968
>>> b[0] = 1
>>> print(b, id(b))
[1] 2882898544968
```

Tuple

Xem
https://www.programiz.com/python-programming/tuple

Tuple vs. List: https://www.programiz.com/python-programming/list-vs-tuples

String

```
>>> # all of the following are equivalent
>>> my_string = 'Hello'
>>> print(my_string)
Hello
>>> my_string = "Hello"
>>> print(my_string)
Hello
>>> my_string = '''Hello'''
>>> print(my_string)
Hello
```

String

```
>>> # triple quotes string can extend multiple lines
>>> my_string = """Hello, welcome to
...               the world of Python"""
>>> print(my_string)
Hello, welcome to
           the world of Python
```

String - Truy cập

```
>>> str = 'programiz'
>>> print('str = ', str)
str =  programiz
>>> print('str[0] = ', str[0])
str[0] =  p
>>> print('str[-1] = ', str[-1])
str[-1] =  z
>>> #slicing 2nd to 5th character
>>> print('str[1:5] = ', str[1:5])
str[1:5] =  rogr
>>> #slicing 6th to 2nd last character
>>> print('str[5:-2] = ', str[5:-2])
str[5:-2] =  am
```

String - Thay đổi/xóa

String là loại immutable, các phép thay đổi tương tự như list sẽ báo lỗi

```
>>> my_string = 'programiz'
>>> my_string[5] = 'a'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> del my_string[1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object doesn't support item deletion
```

String - phép tính

```
>>> str1 = 'Hello'
>>> str2 ='World!'
>>> # using +
>>> print('str1 + str2 = ', str1 + str2)
str1 + str2 =  HelloWorld!
>>> # using *
>>> print('str1 * 3 =', str1 * 3)
str1 * 3 = HelloHelloHello
```

String

```
>>> str = 'cold'
>>> # membership
>>> 'c' in str
True
>>> 'a' in str
False
>>> # enumerate()
>>> list_enumerate = list(enumerate(str))
>>> print('list(enumerate(str) = ', list_enumerate)
list(enumerate(str) =  [(0, 'c'), (1, 'o'), (2, 'l'), (3, '
>>> #character count
>>> print('len(str) = ', len(str))
len(str) =  4
```

String

Xem thêm nhiều nội dung khác tại

https://www.programiz.com/python-programming/string

Set

Xem https://www.programiz.com/python-programming/set

Set vs. list: mỗi phần tử của set là duy nhất và immutable. Tuy nhiên bản thân set thì là mutable, ta có thể thêm hoặc xóa phần tử từ nó.

```
>>> my_list = [1,2,3,1,2]
>>> print(my_list)
[1, 2, 3, 1, 2]
>>> my_tuple = (1,2,3,1,2)
>>> print(my_tuple)
(1, 2, 3, 1, 2)
>>> my_set = {1,2,3,1,2}
>>> print(my_set)
{1, 2, 3}
```

Dictionary

Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a key: value pair.

www.programiz.com/python-programming/dictionary

```
>>> # empty dictionary
>>> my_dict = {}
>>> # dictionary with integer keys
>>> my_dict = {1: 'apple', 2: 'ball'}
>>> # dictionary with mixed keys
>>> my_dict = {'name': 'John', 1: [2, 4, 3]}
>>> # using dict()
>>> my_dict = dict({1:'apple', 2:'ball'})
>>> # from sequence having each item as a pair
>>> my_dict = dict([(1,'apple'), (2,'ball')])
```

Dictionary - Truy cập phần tử

```
>>> my_dict = {'name':'Jack', 'age': 26}
>>> # Output: Jack
>>> print(my_dict['name'])
Jack
>>> # Output: 26
>>> print(my_dict.get('age'))
26
>>> # Trying to access keys which doesn't exist throws erro
>>> # my_dict.get('address')
>>> # my_dict['address']
```

Dictionary - Thay đổi

```
>>> my_dict = {'name':'Jack', 'age': 26}
>>> # update value
>>> my_dict['age'] = 27
>>> print(my_dict)
{'name': 'Jack', 'age': 27}
>>> # add item
>>> my_dict['address'] = 'Downtown'
>>> print(my_dict)
{'name': 'Jack', 'age': 27, 'address': 'Downtown'}
```

Dictionary - Xóa

```
>>> squares = {1:1, 2:4, 3:9, 4:16, 5:25}
>>> print(squares.pop(4))  # remove a particular item
16
>>> print(squares)
{1: 1, 2: 4, 3: 9, 5: 25}
>>> del squares[5] # delete a particular item
>>> print(squares)
{1: 1, 2: 4, 3: 9}
>>> squares.clear() # remove all items
>>> print(squares)
{}
>>> # delete the dictionary itself
>>> del squares
```

Dictionary

Xem thêm các nội dung tại https://www.programiz.com/python-programming/dictionary

Slicing

https://python-reference.readthedocs.io/en/latest/docs/brackets/slicing.html

Biến đổi giữa các loại dữ liệu

Tự đọc https://www.programiz.com/python-programming/type-conversion-and-casting

Phép toán

Xem www.programiz.com/python-programming/operators

```
>>> x = 14
>>> y = 4
>>> print('x + y =',x+y, '   ', 'x - y =',x-y)
x + y = 18     x - y = 10
>>> print('x * y =',x*y)
x * y = 56
>>> print('x / y =',x/y)
x / y = 3.5
>>> print('x // y =',x//y)
x // y = 3
>>> print('x % y =',x%y)
x % y = 2
>>> print('x ** y =',x**y)
x ** y = 38416
```

Phép so sánh

```
>>> x = 10
>>> y = 12
>>> print('x > y  is',x>y)
x > y  is False
>>> print('x < y  is',x<y)
x < y  is True
>>> print('x == y is',x==y)
x == y is False
>>> print('x != y is',x!=y)
x != y is True
>>> print('x >= y is',x>=y)
x >= y is False
>>> print('x <= y is',x<=y)
x <= y is True
```

Phép toán logic

```
>>> x = True
>>> y = False

>>> print('x and y is',x and y)
x and y is False
>>> print('x or y is',x or y)
x or y is True
>>> print('not x is',not x)
not x is False
```

Phép gán

```
>>> x = 2
>>> x
2
>>> x += 5
>>> x
7
>>> x -= 5
>>> x
2
>>> x *= 5
>>> x
10
>>> x **= 5
>>> x
100000
```

Các phép toán khác

Xem www.programiz.com/python-programming/operators

Hàm - Function

Mục đích

- Chia nhỏ chương trình thành những phần nhỏ, giúp dễ quản trị, bảo trì, kiểm tra lỗi khi chương trình ngày một lớn.
- Tránh việc code lặp đi lặp lại, có thể tái sử dụng code.

Cú pháp

```
def function_name(parameters):
    """docstring"""
    statements
```

Hàm - Function

```
>>> def greet(name):
...     """This function greets to
...     the person passed in as
...     parameter"""
...     print("Hello, " + name + ". Good morning!")
...
>>> greet("Nam")
Hello, Nam. Good morning!
```

Hàm - Function

```
>>> def add_2(num):
...     return num + 2
...
>>> a = 2
>>> b = add_2(a)
>>> b
4
```

Hàm - Function

```
>>> #default param
>>> def add_n(num, n=3):
...     return num + n
...
>>> a = 2
>>> b = add_n(a)
>>> b
5
>>> b = add_n(a,-2)
>>> b
0
```

Xem thêm

- www.programiz.com/python-programming/function
- file 02_function.py.

Mutable vs immutable function parameter

```
>>> #immutable parameter
>>> def minus_2(num):
...     print(id(num))
...     num -= 2
...     print(num, id(num))
...
>>> a = 1
>>> print(id(a))
140734095733584
>>> minus_2(a)
140734095733584
-1 140734095733520
>>> print('a =', a, ' - id(a)=', id(a))
a = 1  - id(a)= 140734095733584
```

Mutable vs immutable function parameter

```
>>> #mutable parameter
>>> def updateList(list1):
...     list1 += [10]
...
>>> n = [5, 6]
>>> print(id(n))
2882895614984
>>> updateList(n)
>>> print(n)
[5, 6, 10]
>>> print(id(n))
2882895614984
```

Xem thêm: https://medium.com/@meghamohan/mutable-and-immutable-side-of-python-c2145cf72747

Điều kiện

```
>>> grade = 7
>>> if grade >= 8.5:
...     print("A")
... elif grade >= 7:
...     print("B")
... elif grade >= 6:
...     print("C")
... elif grade >=5:
...     print("D")
... else:
...     print("E")
...
B
```

Điều kiện

```
>>> a = 3
>>> b = 5
>>> c = a==b
>>> c
False
>>> d = a>b
>>> d
False
>>> e = a!=b
>>> e
True
>>> f = 2<a<4
>>> f
True
```

Vòng lặp for

```
>>> # loop
>>> for i in [1,2,5,7]:
...     print(i)
...
1
2
5
7
```

Vòng lặp for

```
>>> # loop
>>> for i in range(1,7):
...     print(i)
...
1
2
3
4
5
6
```

Vòng lặp for

```
>>> n = 3
>>> # range(start,end) = [start,end)
>>> for i in range(1,n):
...     print(i)
...
1
2
>>> # range(end) = range(0,end)
>>> for i in range(n):
...     print(i)
...
0
1
2
```

Vòng lặp for: lệnh break

```
>>> # break
>>> for i in range(1,10):
...     if i==5:
...         break
...     print(i)
...
1
2
3
4
```

Vòng lặp for: lệnh continue

```
>>> # continue
>>> for i in range(1,10):
...     if i==5:
...         continue
...     print(i)
...
1
2
3
4
6
7
8
9
```

Vòng lặp trên list

```
>>> # loop over list
>>> animals = ["dog","cat","bird"]
>>> for i in animals:
...     print(i)
...
dog
cat
bird
>>> # what if we need indexes?
>>> for index, value in enumerate(animals):
...     print(index, value)
...
0 dog
1 cat
2 bird
```

Vòng lặp trên dictionary

```
>>> # loop over dictionaries
>>> age = {}
>>> age['Tuan'] = 22
>>> age['Lan'] = 18
>>> age['Huong'] = 17
>>> for i in age.keys():
...     print(i,age[i])
...
Tuan 22
Lan 18
Huong 17
```

Vòng lặp trên dictionary

```
>>> # loop over dictionaries
>>> age = {}
>>> age['Tuan'] = 22
>>> age['Lan'] = 18
>>> age['Huong'] = 17
>>> for value in age.values():
...     print(value)
...
22
18
17
```

Vòng lặp trên dictionary

```
>>> # loop over dictionaries
>>> age = {}
>>> age['Tuan'] = 22
>>> age['Lan'] = 18
>>> age['Huong'] = 17
>>> for i,value in age.items():
...     print(i, value)
...
Tuan 22
Lan 18
Huong 17
```

Vòng lặp while

```
>>> animals = ["dog","cat","bird"]
>>> j = 0
>>> while j<len(animals):
...     print(j,animals[j])
...     j += 1
...
0 dog
1 cat
2 bird
```

Hàm range

```
>>> # range(start,end)
>>> list(range(1,5))
[1, 2, 3, 4]
>>> # range(end) = range(0,end)
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> # range(start,end,k)
>>> # range(start,end) = range(start,end,1)
>>> list(range(1,5,3))
[1, 4]
```

Hàm range

```
>>> # range from 0 to 5 with step size 3?
>>> # this will give empty list start=5, end=3
>>> list(range(5,3))
[]
>>> # this is OK
>>> list(range(0,5,3))
[0, 3]
```

Vòng lặp

Tham khảo thêm

- www.programiz.com/python-programming/for-loop
- www.programiz.com/python-programming/looping-technique
- www.southampton.ac.uk/~fangohr/teaching/python/book/html/06-control-flow.html
- File 04_iteration.py.

Lệnh xử lí file

Xem

www.programiz.com/python-programming/file-operation