# LẬP TRÌNH PYTHON CƠ BẢN

## (Basic Python Programming)

Th.S Nguyễn Hoàng Thành

Email: thanhnh@ptithcm.edu.vn

Tel: 0909 682 711

# DANH SÁCH

# 1. ĐẶT VẤN ĐỀ

# Bài toán 1

- Viết chương trình quản lý chuỗi cửa hàng của một thương hiệu thời trang gồm các công việc chính sau (Biết rằng chuỗi cửa hàng này gồm 10 cửa hàng):
  - Nhập doanh số bán hàng từ mỗi cửa hàng
  - Hiển thị cửa hàng có doanh số bán hàng tốt nhất.
  - Tìm kiếm cửa hàng có doanh số bán hàng tốt nhất

# Bài toán 1 (tt)

- Để viết chương trình ta cần các việc:
  - Sử dụng 10 biến để lưu trữ doanh số của 10 cửa hàng.
  - Sử dụng các phép toán so sánh để tìm ra cửa hàng có doanh số bán hàng cao nhất, thấp nhất và sắp xếp giá trị các biến.
  - Sử dụng lệnh print() để hiển thị kết quả.

# Bài toán 1 (tt)

- Nhập doanh số của 10 cửa hàng

```
doanh_so_ch1 = int(input("Nhập doanh số bán hàng của CH 1: "))
doanh_so_ch2 = int(input("Nhập doanh số bán hàng của CH 2: "))
doanh_so_ch3 = int(input("Nhập doanh số bán hàng của CH 3: "))
doanh_so_ch4 = int(input("Nhập doanh số bán hàng của CH 4: "))
doanh_so_ch5 = int(input("Nhập doanh số bán hàng của CH 5: "))
doanh_so_ch6 = int(input("Nhập doanh số bán hàng của CH 6: "))
doanh_so_ch7 = int(input("Nhập doanh số bán hàng của CH 7: "))
doanh_so_ch8 = int(input("Nhập doanh số bán hàng của CH 8: "))
doanh_so_ch9 = int(input("Nhập doanh số bán hàng của CH 9: "))
doanh_so_ch10 = int(input("Nhập doanh số bán hàng của CH 10: "))
```

# Bài toán 1 (tt)

- Viết chương trình tìm kiếm cửa hàng có doanh thu cao nhất

```
doanh_so_max = doanh_so_ch1
if doanh_so_max<doanh_so_ch2:
    doanh_so_max = doanh_so_ch2
if doanh_so_max<doanh_so_ch3:
    doanh_so_max = doanh_so_ch3
…
if doanh_so_max<doanh_so_ch8:
    doanh_so_max = doanh_so_ch8
if doanh_so_max<doanh_so_ch9:
    doanh_so_max = doanh_so_ch9
if doanh_so_max<doanh_so_ch10:
    doanh_so_max = doanh_so_ch10
print('Doanh số bán hàng cao nhất là:', doanh_so_max)
```

# Bài toán 1 (tt)

- Viết chương trình tìm kiếm cửa hàng có doanh thu cao nhất

```
print('Các cửa hàng có doanh số cao nhất là')
if doanh_so_max == doanh_so_ch1:
    print("Cửa hàng 1")
if doanh_so_max == doanh_so_ch2:
    print("Cửa hàng 2")
if doanh_so_max == doanh_so_ch3:
    print("Cửa hàng 3")
…
if doanh_so_max == doanh_so_ch8:
    print("Cửa hàng 8")
if doanh_so_max == doanh_so_ch9:
    print("Cửa hàng 9")
if doanh_so_max == doanh_so_ch10:
    print("Cửa hàng 10")
```

# 2. DANH SÁCH

# 2.1 Định nghĩa

- Danh sách là một thùng chứa một hoặc nhiều phần tử, các phần tử trong một danh sách có thể có các giá trị thuộc các kiểu dữ liệu khác nhau, danh sách gồm một số đặc điểm sau:
  - Các phần tử trong danh sách được đặt trong cặp ngoặc vuông [].
  - Các phần tử trong danh sách được phân cách nhau bởi dấu phẩy.
  - Danh sách có khả năng chứa mọi giá trị, đối tượng trong Python.
  - Danh sách có thể chứa một danh sách khác.

# Python - Lists

- Lists are Python's most flexible ordered collection object type.
- Lists can contain any sort of object:
  - numbers,
  - strings,
  - and even other lists.
- Lists may be changed in place by assignment to offsets and slices, list method calls, deletion state- ments, and more—they are ***mutable*** objects.

# Python lists are:

- *Ordered collections of arbitrary objects*
- *Accessed by offset*
- *Variable-length, heterogeneous, and arbitrarily nestable*
- *Of the category "mutable sequence"*
- *Arrays of object references*

# Common list literals and operations (Table 8.1, p.248 – 249)

| Operation | Interpretation |
|---|---|
| L = [] | An empty list |
| L = [123, 'abc', 1.23, {}] | Four items: indexes 0..3 |
| L = ['Bob', 40.0, ['dev', 'mgr']] | Nested sublists |
| L = list('spam') | List of an iterable's items, list of successive integers |
| L = list(range(-4, 4)) | |
| L[i] | Index, index of index, slice, length |
| L[i][j] | |
| L[i:j] | |
| len(L) | |

# Common list literals and operations (Table 8.1, p.248 – 249)

| Operation | Interpretation |
|---|---|
| L1 + L2 | Concatenate, repeat |
| L*3 | |
| for x in L: print(x) | Iteration, membership |
| 3 in L | |
| L.append(4) | Methods: growing |
| L.extend([5,6,7]) | |
| L.insert(i, X) | |
| L.index(X) | Methods: searching |
| L.count(X) | |

# Common list literals and operations (Table 8.1, p.248 – 249)

| Operation | Interpretation |
|---|---|
| L.sort() | Methods: sorting, reversing, copying (3.3+), clearing (3.3+) |
| L.reverse() | |
| L.copy() | |
| L.clear() | |
| L.pop(i) | Methods, statements: shrinking |
| L.remove(X) | |
| del L[i] | |
| del L[i:j] | |
| L[i:j] = [] | |

# Common list literals and operations (Table 8.1, p.248 – 249)

| Operation | Interpretation |
|---|---|
| L[i] = 3 | Index assignment, slice assignment |
| L[i:j] = [4,5,6] | |
| L = [x**2 for x in range(5)] | List comprehensions and maps |
| list(map(ord, 'spam')) | |

# Ví dụ 1:

```
>>> a = [1, 2, 3, 4, 5]
>>> b = ['a', 'b', 'c', 'd']
>>> c = [[1, 2], [3, 4]]
>>> d = [1, 4.0, 'one', [2, 'two']]
```

# 2.2 Khởi tạo danh sách

Để tạo danh sách trong Python ta có thể sử dụng một trong các cách sau:

- **Cách 1: Tạo danh sách dùng lệnh gán**
  - **Cú pháp:** *<tên biến> = [<giá trị 1>, <giá trị 2>, ..., <giá trị n>]*
  - **Trong đó:**
    - *<tên biến> là tên của một biến do người dùng đặt*
    - *<giá trị 1>, <giá trị 2>,...<giá trị n> là dãy các phần tử khởi đầu được lưu trữ trong danh sách và được quản lý bởi <tên biến>. Nếu không đưa giá trị vào thì danh sách là rỗng.*

# Ví dụ 2:

- Tạo danh sách lưu trữ một thông tin sinh viên như sau:

```
>>> sv = ["Lê Thị Hoa", 21, "141, Lý Chính
 Thắng, P7, Q.3, TP.HCM", 8.0, 6.0, 7.0]

>>> print("sv = ", sv)

sv =  ['Lê Thị Hoa', 21, '141, Lý Chính Th
ắng, P7, Q.3, TP.HCM', 8.0, 6.0, 7.0]

>>> print(type(sv))
<class 'list'>
```

# Ví dụ 3:

- Khai báo và khởi tạo giá trị cho danh sách

```
>>> lst1 = []
>>> print("lst1 = ", lst1)
lst1 =  []
>>> lst2 = [0, 1, 2, 3, 4, 5, 6]
>>> print("lst2 = ", lst2)
lst2 =  [0, 1, 2, 3, 4, 5, 6]
>>> lst3 = [1, 2, 3, [1, 2, 5], "Python"]
>>> print("lst3 = ", lst3)
lst3 =  [1, 2, 3, [1, 2, 5], 'Python']
```

# Ví dụ 3 (tt):

- Khai báo và khởi tạo giá trị cho danh sách

```
>>> lst4 = [item for item in range(3)]
>>> print(lst4)

[0, 1, 2]

>>> lst5 = [[n, n * 1, n * 2] for n in range(1, 4)]
>>> print("lst5 = ", lst5)

lst5 =  [[1, 1, 2], [2, 2, 4], [3, 3, 6]]


>>> lst6 = [x for x in range(0, 10) if x > 2 and
x < 6 and x != 5]
>>> print("lst6 = ", lst6)

lst6 =  [3, 4]
```

# 2.2 Khởi tạo danh sách (tt)

**Cách 2:**

- Sử dụng hàm list() để tạo danh sách.
- Cú pháp: *list(col)*
- Trong đó:
  - col có thể là một xâu ký tự, một list, tuple, hoặc set; nếu ta không truyền vào đối số cho list thì hàm trả về một danh sách rỗng.

# Ví dụ 4:

```
>>> lst = list()
>>> print("lst = ", lst)
lst =  []

>>> print(type(lst))
<class 'list'>
```

# 2.3 Truy cập đến các phần tử trong danh sách

- Để truy cập chúng ta dùng biến (variable) và chỉ số (index)
- Cú pháp:

*<tên biến>[<chỉ số>]*

- Trong đó:
  - *<tên biến>:* là tên của danh sách
  - *<chỉ số>:* Nhận các giá trị từ -n đến n-1, với n là số phần tử trong danh sách.

## 2.3 Truy cập đến các phần tử trong danh sách (tt)

❖ Để lấy về một danh sách con chứa các phần tử liên tục trong danh sách, ta thực hiện cú pháp:

*<tên biến>[<start:end:step>]*

❖ Trong đó:

- *<tên biến>*: là tên của danh sách
- *<start>:* là một giá trị số nguyên chỉ vị trí bắt đầu lấy trong danh sách, giá trị mặc định là 0.
- *<end>:* là một giá trị số nguyên chỉ vị trí bắt đầu lấy trong danh sách, giá trị mặc định là n-1.
- *<step>:* Là bước nhảy, giá trị mặc định là 1

# Ví dụ 5:

```
>>> L = [1, 2, 3, 4, 5, 6]
>>> print(L[0])
1
>>> print(L[-5])
2
>>> print(L[5])
6
>>> print(L[-2])
5
```

# Ví dụ 5 (tt)

```
>>> L = [1, 2, 3, 4, 5, 6]
>>> print(L[1: -1])
[2, 3, 4, 5]
>>> print(L[1: 0])
[]
>>> print(L[1: 3])
[2, 3]
```

# 2.4 Thêm phần tử vào danh sách

- Để thêm một phần tử vào danh sách, chúng ta có thể sử dụng một trong các phương thức sau:
  - append(x): Thêm phần tử có giá trị x vào cuối danh sách.
  - insert(i, x): Thêm phần tử có giá trị x tại vị trí có chỉ số i trong danh sách.
    - Nếu i = 0 hoặc i <= -n thì x sẽ được thêm vào đầu danh sách.
    - Nếu i > n -1 thì x sẽ được thêm vào cuối danh sách. Với n là số phần tử của danh sách.

# Ví dụ 6:

```
>>> L = [1]
>>> L.append(2)
>>> print(L)
[1, 2]
>>> L.insert(0, 3)
>>> print(L)
[3, 1, 2]
>>> L.insert(5, 4)
>>> print(L)
[3, 1, 2, 4]
```

# Ví dụ 7

- Đoạn chương trình nhập doanh số bán hàng tại cửa hàng có thể viết lại như sau:

```
lst_dsch = []
for i in range(1, 11):
    doanh_so = int(input("Nhập doanh số cửa hàng thứ "+str(i)+":"))
    lst_dsch.append(doanh_so)
print(lst_dsch)
```

# Ví dụ 7 (tt)

Nhập doanh số cửa hàng thứ 1: 100
Nhập doanh số cửa hàng thứ 2: 101
Nhập doanh số cửa hàng thứ 3: 102
Nhập doanh số cửa hàng thứ 4: 103
Nhập doanh số cửa hàng thứ 5: 104
Nhập doanh số cửa hàng thứ 6: 100
Nhập doanh số cửa hàng thứ 7: 105
Nhập doanh số cửa hàng thứ 8: 106
Nhập doanh số cửa hàng thứ 9: 107
Nhập doanh số cửa hàng thứ 10: 95
[100, 101, 102, 103, 104, 100, 105, 106, 107, 95]

# 2.5 Xóa phần tử trong danh sách

❖ Để thực hiện xóa một phần tử trong danh sách, chúng ta có thể xóa theo chỉ số (vị trí) của phần tử trong danh sách hoặc xóa theo giá trị phần tử.

❖ Xóa theo chỉ số của phần tử trong danh sách:

❖ Cú pháp: *<tên biến>._delitem_(<index>)*

❖Trong đó:

- *<Tên biến>*: Là tên danh sách do người dùng đặt.
- *<index>*: chỉ số, vị trí của phần tử trong danh sách muốn xóa, phần tử đầu tiên trong danh sách có chỉ số bằng 0 hoặc –n. Nếu index thuộc [-n, n-1] thì câu lệnh không làm gì và báo lỗi.

# Ví dụ 8

```
>>> L = [1, 2, 3, 4, 5]
>>> print("danh sách L chưa xóa = ", L)
danh sách L chưa xóa =  [1, 2, 3, 4, 5]

>>> L.__delitem__(1)
>>> print("danh sách L đã xóa = ", L)
danh sách L đã xóa =  [1, 3, 4, 5]
```

# 2.5 Xóa phần tử trong danh sách (tt)

- Xóa theo giá trị của phần tử trong danh sách: Ta sử dụng phương thức **remove()** của lớp danh sách.
- *<tên biến>.remove(x):* Xóa phần tử có giá trị bằng x ra khỏi danh sách *<tên biến>*.
  - Nếu trong danh sách có nhiều giá trị x thì giá trị x đầu tiên trong danh sách sẽ được xóa.
  - Nếu trong danh sách không có bất kỳ giá trị x nào thì sẽ phát sinh ngoại lệ "ValueError: list.remove(x): x not in list" và chương trình sẽ bị dừng lại.

# Delete List Elements

- We can delete one or more items from a list using the Python ***del* statement**. It can even delete the list entirely.
- We can use **remove()** to remove the given item or pop() to remove an item at the given index.
- The **pop()** method removes and returns the last item if the index is not provided. This helps us implement lists as stacks (first in, last out data structure).
- And, if we have to empty the whole list, we can use the **clear()** method.
- Finally, we can also delete items in a list by assigning an empty list to a slice of elements.

# Ví dụ 9

```
>>> L = [1, 2, 3, 2, 4]
>>> print("danh sách L chưa xóa = ", L)
danh sách L chưa xóa =  [1, 2, 3, 2, 4]

>>> L.remove(2)
>>> print("danh sách L đã xóa = ", L)
danh sách L đã xóa =  [1, 3, 2, 4]

>>> L.remove(2)
>>> print("danh sách L đã xóa = ", L)
danh sách L đã xóa =  [1, 3, 4]

>>> L.remove(2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x not in list
```

# 2.6 Duyệt phần tử trong danh sách

- Để duyệt qua các phần tử trong danh sách chúng ta sử dụng cấu trúc lặp for hoặc while.
- **Ví dụ 10:** Duyệt các phần tử trong danh sách sinh viên sv và hiển thị ra màn hình

```
>>> sv = ["Lê Thị Hoa", 21, "TP.Hồ Chí Minh"]
>>> for i in sv:
...     print(i)
...

Lê Thị Hoa
21
TP.Hồ Chí Minh
```

# 2.6 Duyệt phần tử trong danh sách (tt)

- **Ví dụ 11:** Duyệt qua tất cả các phần tử trong danh sách sinh viên sv cùng chỉ số và hiển thị kết quả ra màn hình.

```
>>> sv = ["Lê Thị Hoa", 21, "TP.Hồ Chí Minh"]
>>> for i, xi in enumerate(sv):
...     print("i = ", i, ", xi = ", xi)
...

i =  0 , xi =  Lê Thị Hoa
i =  1 , xi =  21
i =  2 , xi =  TP.Hồ Chí Minh
```

# 2.7 Một số phương thức khác

- **len(<list>)**: Trả về số phần tử trong danh sách.
- **sum(<list>)**: Trả về tổng giá trị các phần tử trong danh sách.
- **sorted(<list>)**: Trả về một danh sách chứa các phần tử trong danh sách đã được sắp xếp theo thứ tự tăng dần về giá trị.
- **sorted(list, reverse = True)**: Trả về một danh sách chứa các phần tử trong danh sách được giảm dần về giá trị.
- **max(list)**: Trả về phần tử có giá trị lớn nhất trong danh sách
- **min(list)**: Trả về phần tử có giá trị nhỏ nhất trong danh sách

# List Iteration and Comprehensions

- Lists respond to all the sequence operations we used on strings

```
>>> 3 in [1, 2, 3]                          # Membership
True
>>> for x in [1, 2, 3]:
...         print(x, end=' ')                # Iteration (2.X uses: print x,)
...
1 2 3
```

- List comprehensions are **a way to build a new list** by applying an expression to each item in a sequence (really, in any iterable), and are close relatives to for loops

```
>>> res = [c * 4 for c in 'SPAM']                    # List comprehensions
>>> res
['SSSS', 'PPPP', 'AAAA', 'MMMM']
```

```
>>> res = []
>>> for c in 'SPAM':                    # List comprehension equivalent
...     res.append(c * 4)
...
>>> res
['SSSS', 'PPPP', 'AAAA', 'MMMM']
```

# Changing Lists in Place

- Because lists are mutable, they support operations that change a list object *in place*.
- Python deals only in object references, this distinction between changing an object in place and creating a new object matters;

# Summary

- List is **created** by placing elements inside square brackets **[]**, separated by commas.
- Access List Elements
  - List Index
  - Negative indexing
- List Slicing in Python
- Add/Change List Elements
- Delete List Elements
- Python List Methods: append(), extend(), insert(), remove(), …

# Summary

- List is **created** by placing elements inside square brackets **[]**, separated by commas.

```
>>> myList = [1, 2, 3]                      # list of integers
>>> myList = []                             # empty list
>>> my_list = [1, "Hello", 3.4]             # list with mixed data types
>>> my_list = ["mouse", [8, 4, 6], ['a']]   # nested list
```

# Bài tập

- Viết chương trình đếm độ dài của một danh sách
- Viết chương trình in ra tổng giá trị các phần tử của một danh sách các số nguyên
- Viết chương trình sắp xếp một danh sách các số nguyên theo thứ tự từ bé đến lớn
- Viết chương trình sắp xếp một danh sách các số nguyên theo thứ tự từ lớn đến bé
- Viết chương trình in ra số bé nhất trong danh sách các số nguyên.
- Viết chương trình in ra số lớn nhất trong danh sách các số nguyên.

# 3. TỪ ĐIỂN (DICTIONARY)

# Từ điển

❖ **Từ điển** để lưu trữ tập dữ liệu có cấu trúc, mỗi phần tử trong từ điển lưu trữ bởi một cặp <*key*>:<*value*> (<*khóa*>:<*giá trị*>) với ràng buộc khóa là duy nhất trong từ điển.

❖ Các đặc tính của từ điển:

- Được giới hạn bởi ***cặp ngoặc nhọn {}***, tất cả những gì trong đó là những phần của từ điển
- Các phần tử của từ điển phân cách nhau bởi **dấu phẩy**.
- Mỗi phần tử của từ điển là một cặp ***<khóa>: <giá trị>***
- Ngăn cách giữa các <*khóa*> và <*giá trị*> là dấu **:**
- Các khóa là duy nhất

# 3.1 Tạo từ điển

❖ **Cách 1: Sử dụng lệnh gán**

❑ **Cú pháp:** *<tên biến> = {<khóa 1>:<giá trị 1>, <khóa 2>: <giá trị 2>,... <khóa n> : <giá trị n>*

❑ Trong đó:

- ***<tên biến>***: Là tên của một biến do người dùng đặt theo quy tắc đặt tên
- ***<khóa 1>, <khóa 2>, <khóa n>***: là danh mục các khóa được dùng như chỉ số của phần tử trong từ điển và không được trùng nhau.
- ***<giá trị 1>, <giá trị 2>, … , <giá trị n>***: Là giá trị được đi kèm với <khóa> tương ứng.

# Ví dụ 12

- Tạo một từ điển để lưu thông tin về một môn học gồm các trường thông tin mã môn học, tên môn học, số tín chỉ:

```
>>> mon_hoc = {"Ma" : "13998", "Ten" : "Lập trìn
h căn bản", "soTC" : 3}

>>> print("Thông tin môn học", mon_hoc)

Thông tin môn học {'Ma': '13998', 'Ten': 'Lập tr
ình căn bản', 'soTC': 3}

>>> print(type(mon_hoc))

<class 'dict'>
```

# Ví dụ 13

- Tạo từ điển lưu thông tin các Khoa trong trường

```
>>> khoa = {1: "Kinh tế", 2: "Công nghệ thông tin"}
>>> print("khoa = ",khoa)
khoa =  {1: 'Kinh tế', 2: 'Công nghệ thông tin'}
>>> print(type(khoa))
<class 'dict'>
```

# 3.1 Tạo từ điển (tt)

❑**Cách 2:** Sử dụng hàm dict() để tạo danh sách

- Cú pháp: ***<tên biến> = dict()***

***Ví dụ 14: Sử dụng hàm để khởi tạo từ điển***

```
>>> khoa = dict()
>>> print("khoa = ", khoa)
khoa =  {}
>>> print(type(khoa))
<class 'dict'>
```

# 3.1 Tạo từ điển (tt)

- ***Ví dụ 15:Sử dụng từ điển để khai báo thông tin về 1 người.***

# 3.2 Truy cập đến các phần tử trong từ điển

**Lấy giá trị gắn với khóa trong từ điển**

- **Cú pháp:**
  - *<tên biến>[<khóa>]*
  - *<tên biến>.get(<khóa>)*

# 3.2 Truy cập đến các phần tử trong từ điển (tt)

• Ví dụ 16: Lấy giá trị có khóa bằng name với biến từ điển person.

```
>>> person = {
...     'Ten': 'Nguyễn văn A',
...     'dia chi' : 'TP.HCM',
...     'tuoi' : 19,
...     'gioi tinh' : 'Nam',
...     'tinh trang hon nhan' : 'Độc thân'
... }

>>> print("Tên = ", person['Ten'])
Tên =  Nguyễn văn A
>>> print("Tuổi = ", person.get('tuoi'))
Tuổi =  19
```

# 3.2 Truy cập đến các phần tử trong từ điển (tt)

- Lấy tất cả các khóa trong từ điển
  - Cú pháp: *<tên biến>.keys()*
- Lấy tất cả các giá trị trong từ điển
  - Cú pháp: *<tên biến>.values()*

# Lấy tất cả các khóa trong từ điển (tt)

```
>>> person = {
...      'Ten': 'Nguyễn văn A',
...      'dia chi' : 'TP.HCM',
...      'tuoi' : 19,
...      'gioi tinh' : 'Nam',
...      'tinh trang hon nhan' : 'Độc thân'
... }
>>> print("person = ", person)
person =  {'Ten': 'Nguyễn văn A', 'dia chi': 'TP.HCM
', 'tuoi': 19, 'gioi tinh': 'Nam', 'tinh trang hon n
han': 'Độc thân'}
>>> print("Các khóa =", person.keys())
Các khóa = dict_keys(['Ten', 'dia chi', 'tuoi', 'gio
i tinh', 'tinh trang hon nhan'])
```

# Lấy tất cả các giá trị trong từ điển (tt)

```
>>> person = {
...      'Ten': 'Nguyễn văn A',
...      'dia chi' : 'TP.HCM',
...      'tuoi' : 19,
...      'gioi tinh' : 'Nam',
...      'tinh trang hon nhan' : 'Độc thân'
... }
>>> print("person = ", person)
person =  {'Ten': 'Nguyễn văn A', 'dia chi': 'TP.HCM
', 'tuoi': 19, 'gioi tinh': 'Nam', 'tinh trang hon n
han': 'Độc thân'}

>>> print("Các giá trị =", person.values())
Các giá trị = dict_values(['Nguyễn văn A', 'TP.HCM',
 19, 'Nam', 'Độc thân'])
```

# Duyệt các phần tử trong từ điển

```
>>> for i in person:
...     print(i, ": ", person[i])
...
Ten :  Nguyễn văn A
dia chi :  TP.HCM
tuoi :  19
gioi tinh :  Nam
tinh trang hon nhan :  Độc thân
```

# 3.3 Thêm/Sửa một phần tử trong từ điển

- Cú pháp: ***<tên biến>.[<khóa>] = <giá trị>***
- Trong đó:
  - *<tên biến>:* Là tên một biến từ điển đã được khai báo trước đó.
  - *<khóa>:* tên khóa, nếu không tồn tại thì sẽ tạo mới, nếu đã tồn tại thì <giá trị> sẽ được thêm vào <khóa>
  - *<giá trị>:* giá trị được lưu trữ trong từ điển đi theo khóa và được quản lý bởi *<tên biến>*

# 3.3 Thêm/Sửa một phần tử trong từ điển (tt)

• Ví dụ 17: Sửa địa chỉ và thêm hệ số lương vào biến person.

```
>>> person["dia chi"] = "Đà Nẵng"
>>> person["hsl"] =  6.0

>>> for i in person:
...     print(i, ": ", person[i])
...

Ten :  Nguyễn văn A
dia chi :  Đà Nẵng
tuoi :  19
gioi tinh :  Nam
tinh trang hon nhan :  Độc thân
hsl :  6.0
```

# 3.4 Xóa phần tử trong từ điển

- Cú pháp:
  - *<tên biến>.__delitem__(<khóa>)*
  - *<tên biến>.clear()*

# 3.4 Xóa phần tử trong từ điển (tt)

• Ví dụ 18: Xóa phần tử có khóa là “dia chi” trong biến person.

```
>>> person.__delitem__("dia chi")
>>> for i in person:
...     print(i, ": ", person[i])
...
Ten :  Nguyễn văn A
tuoi :  19
gioi tinh :  Nam
tinh trang hon nhan :  Độc thân
hsl :  6.0
>>> person.clear()
>>> for i in person:
...     print(i, ": ", person[i])
...
>>>
```

# 4. Tạo ứng dụng

- **Ứng dụng 1:** Xây dựng chương trình cho người dùng nhập vào thông tin của một đa thức $P_n = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$
- Phân tích:
  - Hệ số của đa thức là một dãy số, do đó có thể sử dụng danh sách để lưu trữ hệ số khi người dùng nhập thông tin.

# 4. Tạo ứng dụng (tt)

- Ứng dụng 2: Một thương hiệu thời trang bao gồm 10 cửa hàng được đặt trên khắp cả nước. Hãy viết chương trình giúp chủ cửa hàng giải quyết các công việc sau:
  - Nhập doanh số của 10 cửa hàng gửi về
  - Sắp xếp doanh số bán hàng từ bé đến lớn
  - Tính doanh số trung bình của 10 cửa hàng
  - Hiển thị doanh số bán hàng lớn nhất