



TIN HỌC CƠ SỞ

Phần 2: Ngôn ngữ lập trình Python

CHƯƠNG 6

BIẾN VÀ CÁC KIỂU DỮ LIỆU CƠ SỞ

Tin cơ sở (LT): 010100229802 - DHKL16A1HN, - DHKL16A2HN

GV. Cao Diệp Thắng

Mục tiêu chương

Nắm vững:

- ☐ Khái niệm biến (variable).
- ☐ Hiểu và sử dụng được biến (variables)
- ☐ Kỹ năng sử dụng các kiểu dữ liệu cơ sở trong chương trình Python
- ☐ Kỹ năng nhập xuất dữ liệu trên Shell



Nội dung



1

Khái niệm định danh, biến, kiểu dữ liệu

2

Biến (variable) trong python

3

Các kiểu dữ liệu

4

Chuyển đổi kiểu dữ liệu

5

Nhập xuất dữ liệu trên shell

6.1. ĐỊNH DANH (IDENTIFIER)

Định danh(**identifier**) là tên gọi mà lập trình viên sẽ gán cho một thực thể trong Python. Nói một cách đơn giản, định danh là do tên người dùng định nghĩa để đại diện cho các khối xây dựng cơ bản của Python. Định danh có thể là một **biến**, một **hàm**, một **lớp**, một **mô đun** hoặc bất kỳ đối tượng khác.

- Tên được đặt cho biến (***variable***), phương thức/hàm (*function*), lớp (*class*), *module* và các đối tượng khác.
- Việc đặt tên được gọi là định danh (*identifier*)
- Identifier bắt đầu bằng các ký tự A-Z, a-z hoặc dấu gạch dưới “_” (*underscore*), tiếp đó là các ký tự chữ, ký tự số 0- 9
- Identifier có phân biệt chữ hoa chữ thường



Ví dụ 6.1. định

X

x

Spam

spam

spAm

total_of_eggs

Total_Of_Eggs

định danh hợp lệ

Lưu ý

- ☐ Python không sử dụng các ký tự đặc biệt, ký tự trắng, dấu câu như @, #, \$, %,
- ☐ Định danh trong Python không được bắt đầu bằng chữ số. Nếu tạo một mã định danh bắt đầu bằng một chữ số thì chúng ta sẽ gặp lỗi cú pháp.
- ☐ Không sử dụng từ khóa để định danh



Ví dụ 6.2. định danh không hợp lệ

`1_a, 3d, 65x`

`so luong, ti le`

`int, char`

`@time, !y`

bắt đầu bằng chữ số

có ký tự không hợp lệ (dấu cách – space) trong tên
trùng với từ khóa của ngôn ngữ python

dùng ký tự đặc biệt @, !



Lưu ý

- Đôi khi định danh do người dùng đặt gồm nhiều từ, khi đó để dễ đọc nên tách các từ bằng cách sử dụng dấu gạch dưới. Ví dụ định danh **danh_sach_sinh_vien** dễ đọc và dễ hiểu hơn so với định danh **danhsachsinhvien**.
- Định danh nên có tính chất gợi nhớ, ví dụ nếu ta muốn lưu trữ các thông tin về các sinh viên vào một biến nào đó thì biến đó nên được đặt tên là **danh_sach_sinh_vien** hay **ds_sv...** Và ngược lại định danh **danh_sach_sinh_vien** chỉ nên dùng để đặt tên cho những đối tượng liên quan đến sinh viên chứ không nên đặt tên cho các đối tượng chứa thông tin.
- Ngôn ngữ Python phân biệt chữ cái thường và chữ cái hoa trong các định danh, tức là **dinh_danh** khác với **Dinh_danh**.
- Một thói quen của những người lập trình là các hằng thường được đặt tên bằng chữ hoa, các biến, hàm,...thì đặt tên bằng chữ thường. Nếu tên gồm nhiều từ thì nên phân cách các từ bằng dấu gạch dưới.

Ví dụ 6.3

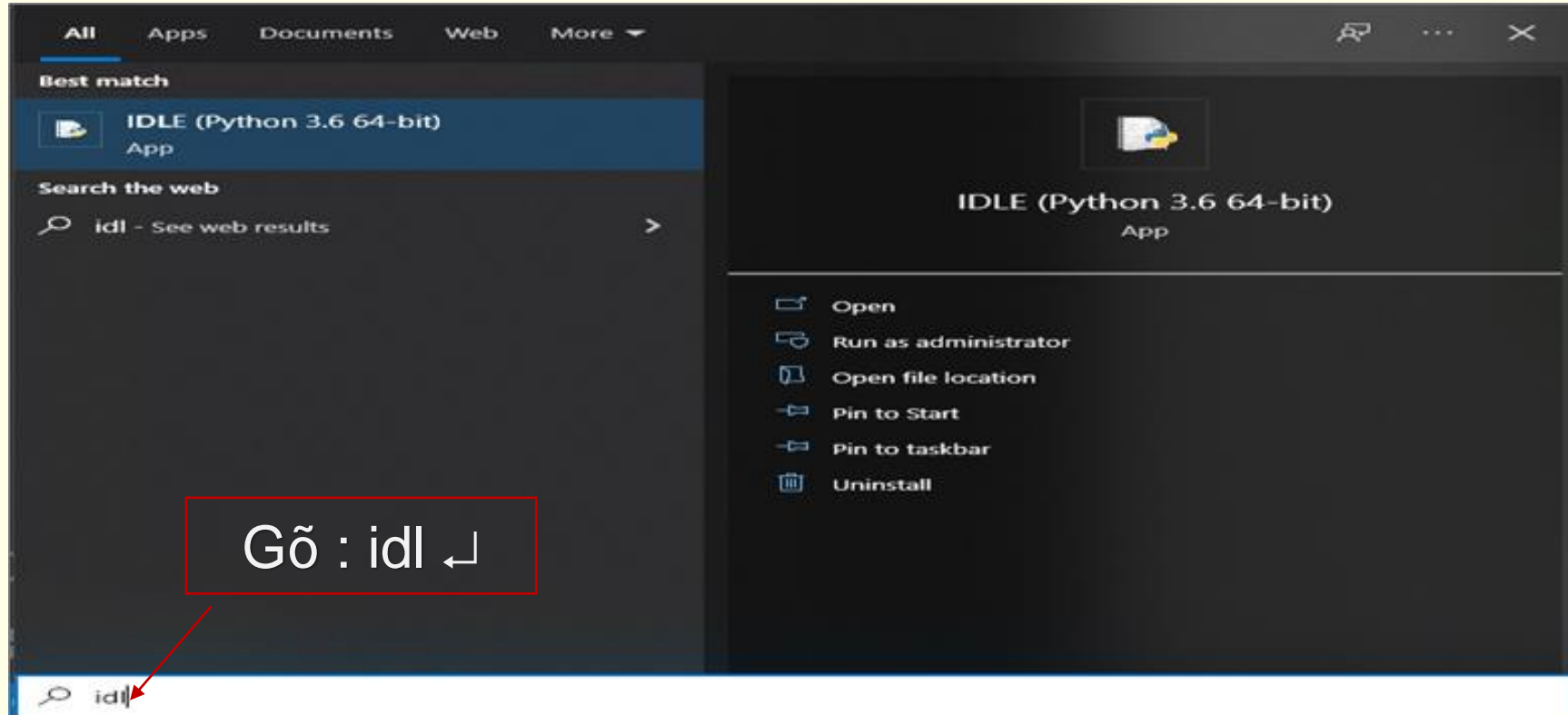
Định danh	Loại đối tượng
HANG_SO_1, CONSTANT_2	hằng
x, y, i, j, count	biến
nhap_du_lieu, tim_kiem, xu_ly	hàm
sinh_vien, nhan_vien, mat_hang	danh sách

6.1.1. Một số qui tắc định danh

- Tên lớp (**class**) bắt đầu bằng chữ hoa. Tất cả các ***identifier*** khác bắt đầu bằng chữ thường
- Tên hàm ***function*** viết thường, các từ nối với nhau bằng dấu _
- Không sử dụng các từ khóa (***keyword***) trong Python khi đặt tên cho bất kỳ *identiffier* nào.

6.1.2. Danh sách từ khóa trong python

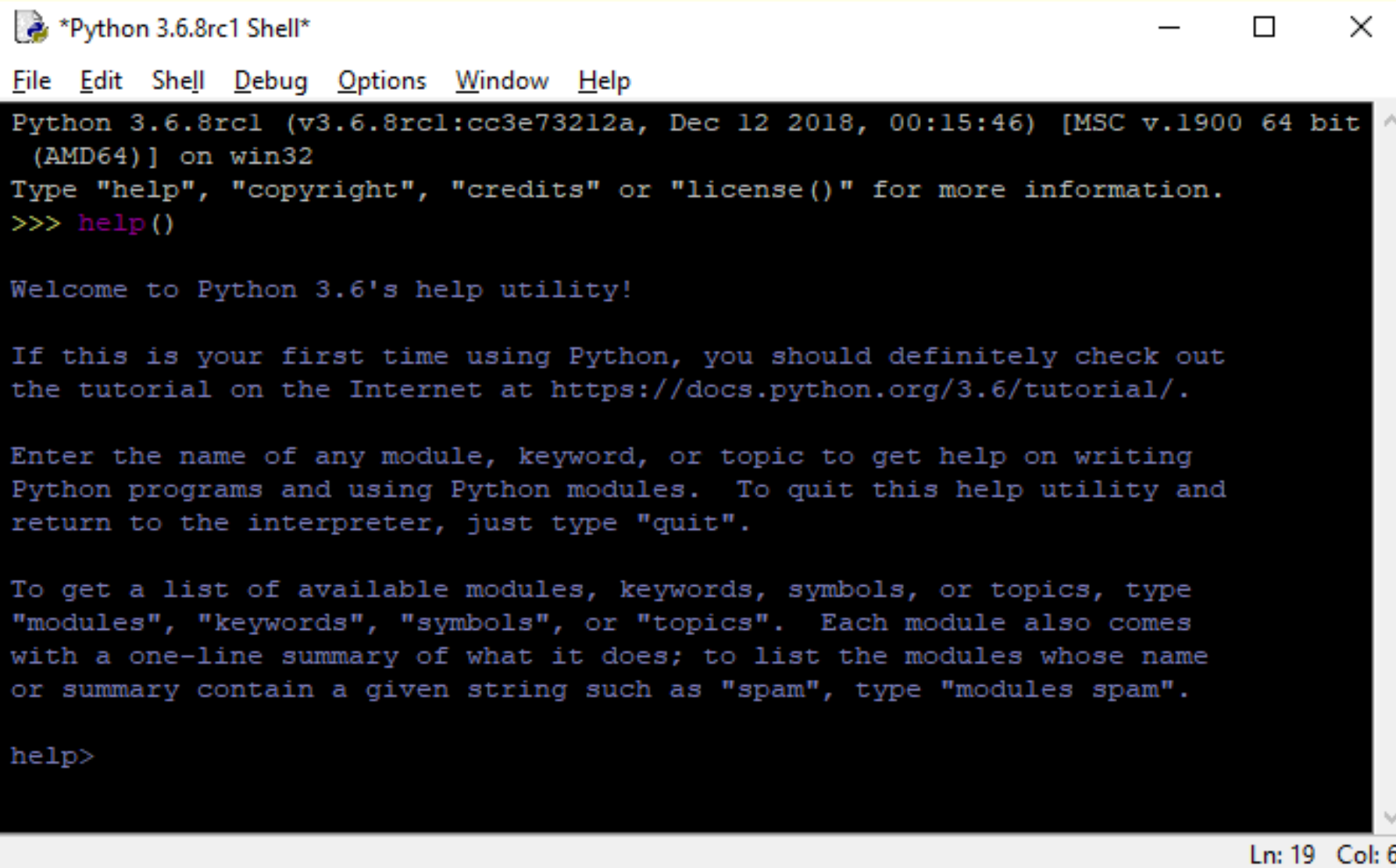
Cách 1. Từ ô tìm kiếm trong windows 7/8/10/11 (Hình 6.1) gõ idl để mở python shell



Hình 6.1. Chuyển sang chế độ Python Shell



Sử dụng lệnh trợ giúp **help()** đã tích hợp trong python, tại dấu nhắc gõ lệnh: **help()** ↵



```
*Python 3.6.8rc1 Shell*
File Edit Shell Debug Options Window Help
Python 3.6.8rc1 (v3.6.8rc1:cc3e73212a, Dec 12 2018, 00:15:46) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> help()

Welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.6/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

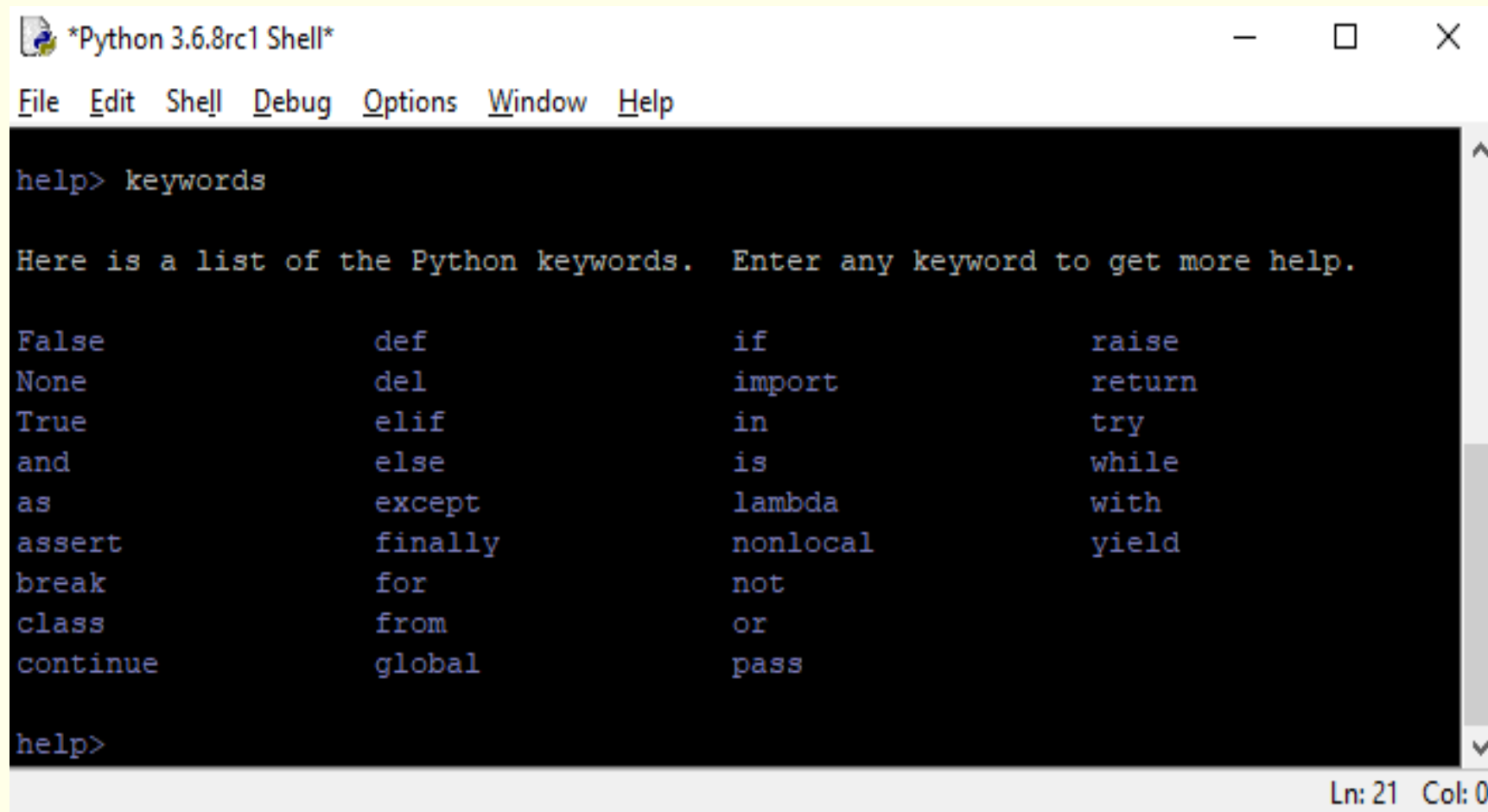
help>
```

Ln: 19 Col: 6

Hình 6.2. Chuyển vào chế độ trợ giúp help()



Tại dấu nhắc help> gõ lệnh: `help>keywords`↵



```
*Python 3.6.8rc1 Shell*
File Edit Shell Debug Options Window Help
help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

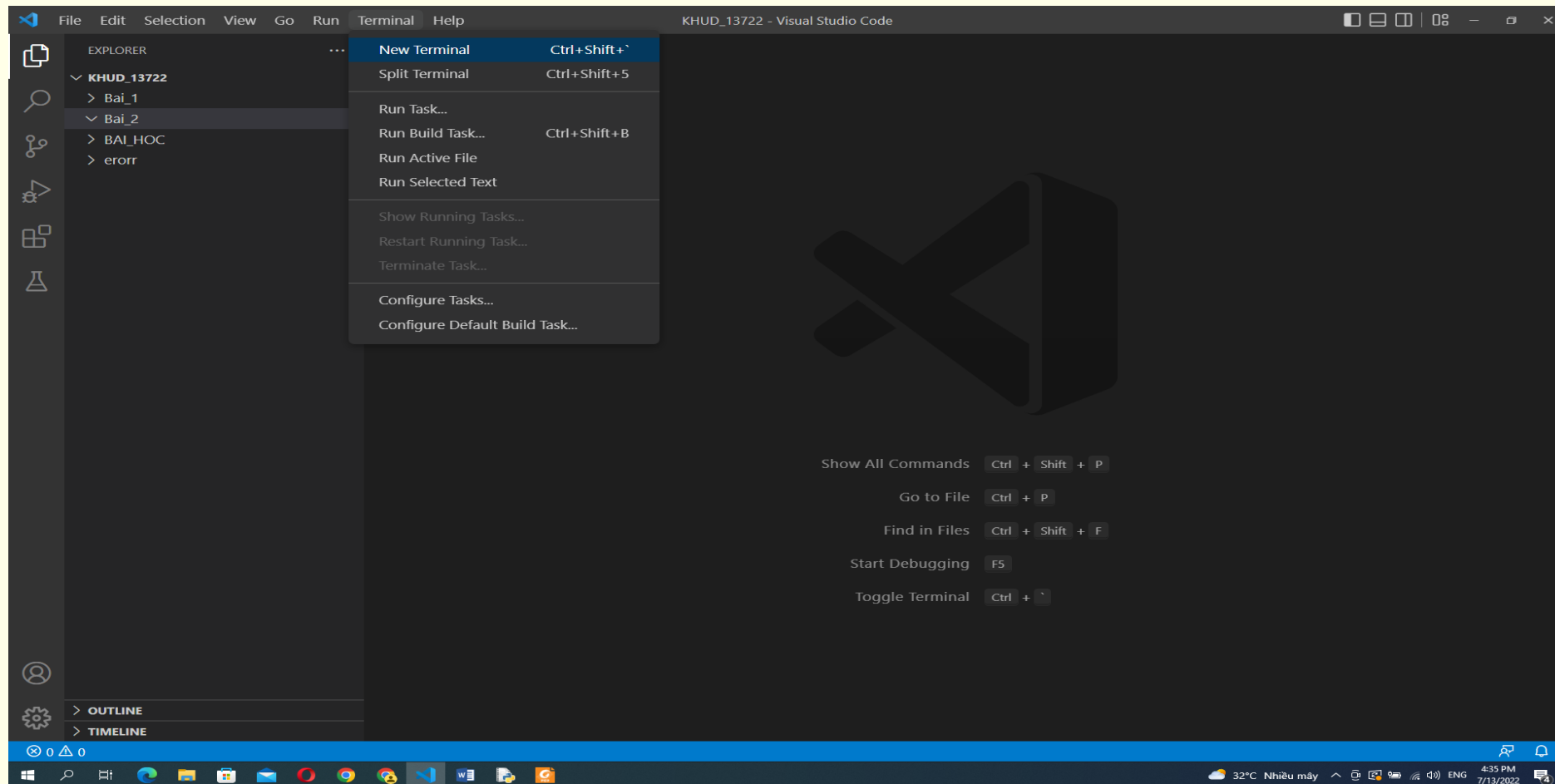
False      def         if          raise
None       del         import      return
True       elif        in          try
and        else        is          while
as         except     lambda     with
assert     finally   nonlocal   yield
break     for        not
class     from       or
continue  global    pass

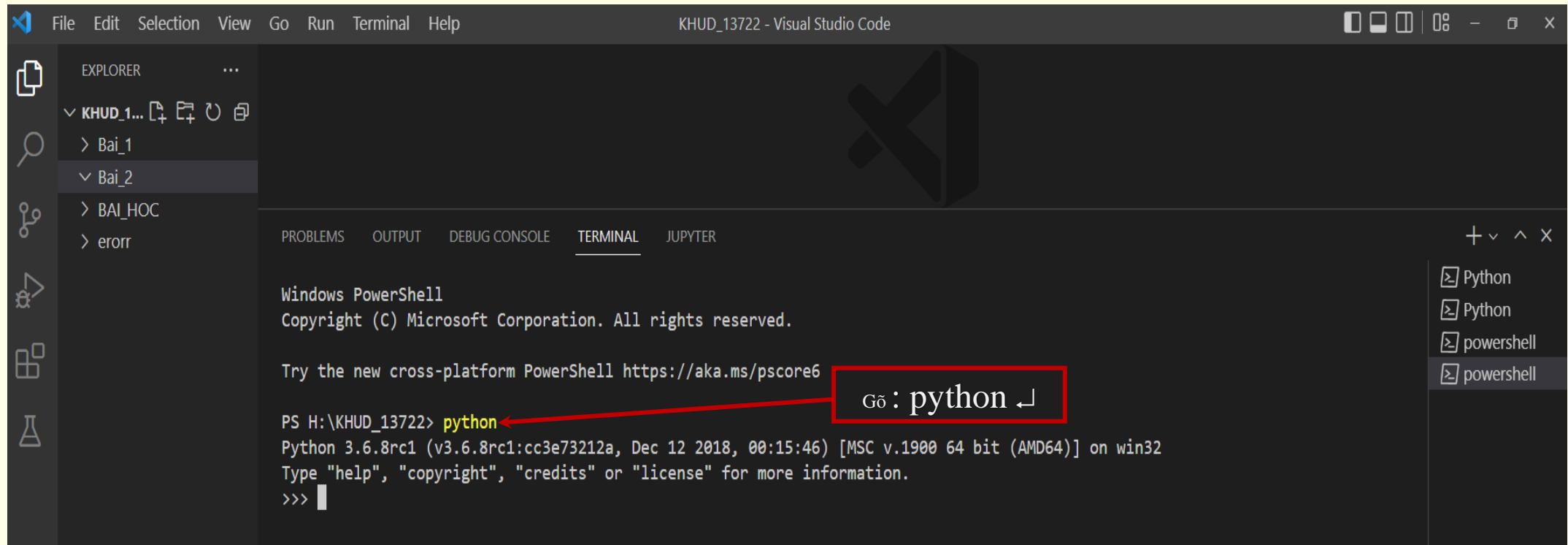
help>
Ln: 21 Col: 0
```



Cách 2.

Trong Visual Studio Code, trên thanh menu chọn Terminal → New Terminal như trên hình 6.4. Xuất hiện cửa sổ Terminal, từ dấu nhắc > gõ: python↵





The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal is running Windows PowerShell. The command prompt shows the user has entered 'python' and the output displays the Python version (3.6.8rc1) and the prompt changes to '>>>'.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS H:\KHUUD_13722> python
Python 3.6.8rc1 (v3.6.8rc1:cc3e73212a, Dec 12 2018, 00:15:46) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

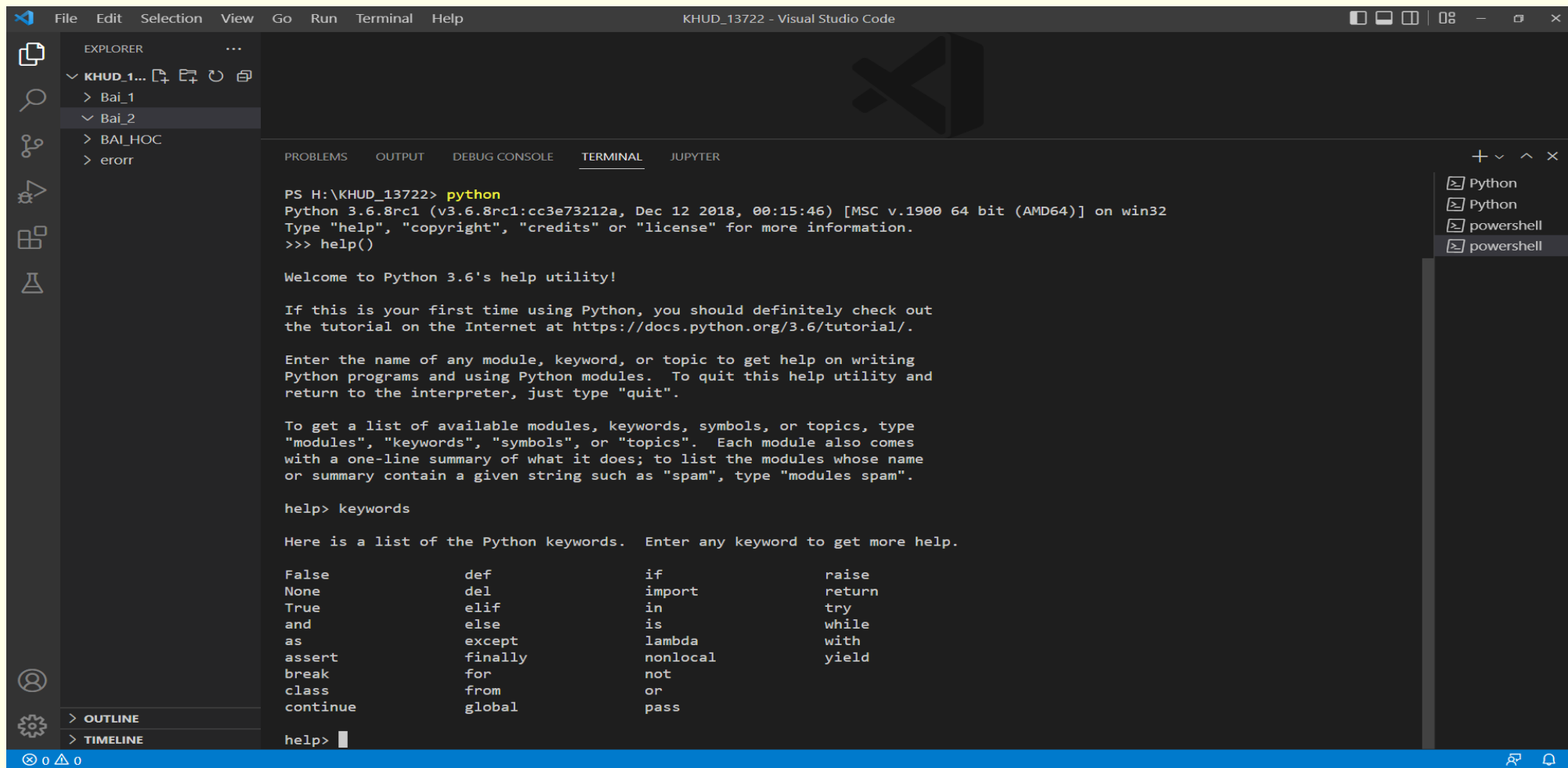
A red box highlights the command 'python' in the terminal, and a red arrow points to it from the text 'Cõ : python'.

Hình 6.5. Chạy chế độ tương tác Python Shell trong VSC



Đến đây lặp lại như cách 1. Kết quả: xem hình 6.6





The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the output of the `python` command, which has entered the Python 3.6 help utility. The user has entered `help> keywords`, and the terminal displays a list of Python keywords arranged in a grid. The Explorer sidebar on the left shows a project structure with folders `Bai_1`, `Bai_2`, `BAI_HOC`, and `errorr`. The bottom status bar shows 0 errors, 0 warnings, and 0 info messages.

```
PS H:\KHUUD_13722> python
Python 3.6.8rc1 (v3.6.8rc1:cc3e73212a, Dec 12 2018, 00:15:46) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> help()

Welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.6/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

False      def         if           raise
None       del         import       return
True       elif        in           try
and        else        is           while
as         except      lambda       with
assert     finally     nonlocal     yield
break      for         not
class      from        or
continue   global      pass

help> 
```

Hình 6.6. Danh mục từ khóa keywords của python

Để thoát khỏi chế độ trợ giúp help, gõ q hoặc ctrl_Z

6.2. BIẾN (VARIABLE) TRONG PYTHON

6.2.1. Khái niệm

- ❑ Trong lập trình, biến (**variable**) là tên của một vùng trong bộ nhớ **RAM**, được sử dụng để lưu trữ thông tin.
- ❑ Lập trình viên có thể gán thông tin cho một biến, và có thể lấy thông tin đó ra để sử dụng.
- ❑ Khi một biến được khai báo, một vùng trong bộ nhớ sẽ dành cho các biến. Các biến được **tham chiếu** trong chương trình để nhận giá trị sau này và giá trị của chúng có thể được thay đổi.

Biến (variable):

- *Là một đơn vị lưu trữ trên bộ nhớ của máy tính, lưu trữ các giá trị có thể được dùng để tính toán xử lý*
- *Biến có thể lưu trữ dữ liệu dạng chuỗi, dạng số, ...*
- *Bằng cách gán các kiểu dữ liệu khác nhau cho biến, ta tạo ra các biến kiểu số nguyên, số thập phân, chuỗi...*
- *Bắt buộc phải khai báo biến khi sử dụng*

6.2.3. Khai báo biến

❑ Cú pháp: **ten_bien** = <gia_tri>

- Trong đó **ten_bien** là tên của biến mà người lập trình muốn đặt,
- **Gia_tri** là giá trị biến mà lập trình viên muốn gán.

Hoặc có thể khai báo nhiều biến cùng một lúc với các giá trị tương ứng:

❑ Cú pháp: biến_1, biến_2, biến_3, ... = <giá_trị_1>, <giá_trị_2>, <giá_trị_3>, ...

Một số lưu ý khi khai báo biến:

- Tên biến chỉ có thể chứa chữ cái, số và dấu gạch dưới `_`. Có thể bắt đầu bằng chữ cái hoặc dấu gạch dưới, nhưng không được bắt đầu bằng số.
- Biến không được chứa khoảng trắng và các ký tự đặc biệt như `+`, `-`, ...
- Biến không được là chữ có dấu (Tiếng Anh)
- Tên biến có phân biệt chữ hoa, chữ thường và không được trùng với các từ khóa của Python như: **and**, **as**, **del**, **from**, ...

Chú ý: Trong Python một biến **không cần khai báo và xác định kiểu dữ liệu trước**, khi gán giá trị thì tự động Python sẽ nội suy ra kiểu dữ liệu của biến.
Như vậy **một biến có thể có nhiều kiểu dữ liệu** tùy thuộc vào giá trị mà ta gán.

Ví dụ 6.4. Khai báo biến trong python

```
name = "Khoa học Dữ liệu"
```

có thể khai báo biến bằng 1 giá trị trên cùng một lần khai báo:

```
a=b=c = 2022
```

khai báo nhiều biến với các giá trị tương ứng của nó trên 1 dòng:

```
name, age, gender = "Nguyen Văn A", 20, True
```

Ví dụ 6.5. Lỗi khi không khai báo biến trước.

```
>>> x
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    x
NameError: name 'x' is not defined
>>>
```

Ví dụ 6.6. Tính tổng, hiệu, tích, thương của hai số nguyên cho trước.

```
Python 3.6.8rc1 Shell
File Edit Shell Debug Options Window Help
Python 3.6.8rc1 (v3.6.8rc1:cc3e73212a, Dec 12 2018, 00:15:46) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> x=5
>>> y=10
>>> tong=x+y
>>> hieu=x-y
>>> tich=x*y
>>> thuong=x/y
>>> print("tong = ", tong)
tong = 15
>>> print("hieu = ",hieu); print("Tich = ", tich); print("Thuong = ", thuong)
hieu = -5
Tich = 50
Thuong = 0.5
>>> khoa="KHOA HOC DU LIEU"
>>> print(khoa)
KHOA HOC DU LIEU
>>> trung_binh=(x+y)/2
>>> print("trung binh cong cua ", x, ", ", y, " = ", trung_binh)
trung binh cong cua 5 , 10 = 7.5
>>>
```

6.2.4. Xóa biến

Có thể xóa một biến bằng lệnh: **del** "tên biến".

Ví dụ 6.7.

```
1 >>> x=5
2 >>> print(x)
3 5
4 >>> del x
5 >>> print(x)
```

Traceback (most recent call last):

File "<pyshell#3>", line 1, in <module>
print(x)

NameError: name 'x' is not defined

Chương trình in ra lỗi do lệnh
del ở dòng 4 đã xóa biến x

6.3. CÁC KIỂU DỮ LIỆU

6.3.1. Khái niệm kiểu dữ liệu

- Dữ liệu là tài nguyên quan trọng nhất trong máy tính, song dữ liệu trong máy tính lại không phải tất cả đều giống nhau.
- Có dữ liệu là chữ viết, có dữ liệu là con số, lại có dữ liệu khác là hình ảnh, âm thanh... Ta nói rằng các dữ liệu đó thuộc các kiểu dữ liệu khác nhau.

Kiểu dữ liệu có thể được định nghĩa gồm 2 điểm như sau:

- Một kiểu dữ liệu là một **tập hợp các giá trị** mà một dữ liệu thuộc kiểu dữ liệu đó có thể nhận được.
- Trên một kiểu dữ liệu ta **xác định một số phép toán** đối với các dữ liệu thuộc kiểu dữ liệu đó.

Lưu ý: ta có thể dùng hàm **type()** để kiểm tra kiểu dữ liệu của biến

Ví dụ 6.8

```
>>> x=5
>>> type(x)
<class 'int'>
>>> y = True
>>> type(y)
<class 'bool'>
```

Trong Python có các kiểu dữ liệu cơ bản và các kiểu dữ liệu danh sách (sẽ trình bày ở các chương sau). .

Kiểu dữ liệu cơ bản là:

- Number - Kiểu số
- Boolean – Kiểu True/ False
- String - Kiểu chuỗi/xâu ký tự

Các kiểu dữ liệu danh sách

Kiểu dữ liệu danh sách có các kiểu:

- List, Tuple, Dictionary

6.3.2. Kiểu dữ liệu Number – Kiểu số

a. Kiểu int

int: là kiểu số nguyên (không có chứa dấu chấm thập phân), có thể lưu các số nguyên âm và dương.

- ❑ Kiểu int biểu diễn các số nguyên có dấu với độ lớn tùy ý.
- ❑ Kiểu int trong Python không sử dụng số bit cố định để biểu diễn như các ngôn ngữ khác chẳng hạn C, C++.
- ❑ Tùy thuộc vào giá trị cụ thể Python sẽ chọn số bit phù hợp. Giá trị số nguyên lớn nhất mà Python biểu diễn được chỉ phụ thuộc vào bộ nhớ.
- Python hỗ trợ biểu diễn số nguyên dương, số nguyên âm, số ở dạng thập phân, hệ cơ số 8, hệ cơ số 16.
- Khi biểu diễn số ở cơ số 8 ta sử dụng tiếp tố **0o** hoặc **0O** (số không và chữ o hoa/thường)
- Khi biểu diễn ở hệ cơ số 16 thì sử dụng tiếp tố **0x** hoặc **0X** (số không và chữ x/X)

Ví dụ 6.9. Biểu diễn số nguyên trong Python

Giá trị	Ghi chú
100	Số nguyên dương
-10	Số nguyên âm
0o100, 0O100	Số dương ở cơ số 8 (số $64_{(10)}$)
-0o100, -0O100	Số âm ở cơ số 8
0x100, 0X100	Số dương ở cơ số 16 ($256_{(10)}$)
-0x100, -0X100	Số âm ở cơ số 16

Python cũng cho phép dùng ký tự gạch chân “_” để nhóm các chữ số khi biểu diễn số:

Ví dụ 6.10.

```
>>> i = 1_000_000 #tuong duong voi 1000000
>>> i
1000000
>>> type(i)
```

a. Kiểu số thực

Trong Python, **float** dùng để biểu diễn số thực dấu phẩy động. Để viết số thực trong Python ta cần đặt 1 dấu chấm thập phân.

Ví dụ 6.11. Biểu diễn số thực trong Python

Giá trị	Ghi chú
0.0	Giá trị 0.0 (float) chưa chắc đã bằng 0 (int)
100.0	Số thực dương
-100	Số thực âm
100e2, -100e2	Cách viết khoa học, bằng $\pm 100 \cdot 10^2 = \pm 10000.0$
100e-2, -100e-2	Cách viết khoa học, bằng $\pm 100 \cdot 10^{-2} = \pm 1.0$
100., -100.	Không cần viết số 0 sau dấu chấm thập phân, tương đương 100.0, -100.0

Python cũng cho phép dùng ký tự gạch chân “_” để nhóm các chữ số khi biểu diễn số thực:

Ví dụ 6.12:

```
>>> j = 1_000_000.001
>>> j
1000000.001
>>> type(j)
<class 'float'>
```

Lưu ý: Trong phép toán mà toán hạng bao gồm cả số thực và số nguyên, kết quả của phép toán sẽ thuộc về kiểu số thực.

Ví dụ 6.13.

```
>>> a = 1
>>> b = 2.0
>>> c = a+b
>>> type(c)
<class 'float'>
```

a. Kiểu số phức (complex)

Python là một trong số ít các ngôn ngữ hỗ trợ trực tiếp kiểu số phức.

Trong toán học, số phức được biểu diễn ở dạng tổng quát $a+bi$ với i là đơn vị ảo. Trong Python, đơn vị ảo được biểu diễn bằng ký tự j hoặc J .

Ví dụ 14. Cách biểu diễn số phức trong Python

Giá trị	Ghi chú
<code>3.14j</code>	Chỉ có phần ảo (<code>0 + 3.14j</code>)
<code>45.j</code>	Hiểu là (<code>0+45j</code>)
<code>3+2j</code>	Số phức (<code>3+2j</code>)
<code>1+3.2e25j</code>	Phần ảo biểu diễn dạng khoa học

Ví dụ 6.15. Xét số phức $z = 2+3j$ thì 2 là phần thực, 3 là phần ảo (j là từ khóa để đánh dấu phần ảo)

Ví dụ 6.16. `z=complex(2,3)` thì 2 là phần thực, 3 là phần ảo khi xuất kết quả ta có thể xuất ra màn hình:

```
print("Phần thực= ", z.real) ==> Phần thực= 2
```

```
print("Phần ảo= ", z.imag) ==> Phần ảo= 3
```

Kết quả:

```
>>> z=2+3j
>>> print("Phần thực = ", z.real)
Phần thực = 2.0
>>> print("Phần ảo = ", z.imag)
Phần ảo = 3.0
```

Lưu ý: khi biểu diễn số phức j (J) phải đi kèm với giá trị số mới được xem là đơn vị ảo. Nếu j hay J đứng độc lập một mình sẽ bị xem là tên biến. Chẳng hạn: $1+j$ là 1 biểu thức (với j là một biến nào đó) nhưng $1+1j$ lại được xem là một số phức.

6.3.3. Kiểu dữ liệu Boolean (logic)

- ❑ Kiểu bool (Boolean) trong Python là kiểu dữ liệu trong đó chỉ có hai giá trị True và False. True và False là hai từ khóa trong Python.

Ví dụ 6.17.

```
>>> result=True
>>> print(result)
True
>>> notresult=not result
>>> print(notresult)
False
```

6.3.4. Kiểu chuỗi/xâu ký tự String

- Là một chuỗi các ký tự được đặt trong nháy kép “” hoặc nháy đơn ‘’
- Khai báo và khởi tạo chuỗi:

`tên_chuỗi = <giá_trị`

Ví dụ 6.18.

```
>>> name = "Ha Noi"
>>> print(name)
Ha Noi
>>> cau = "Ha Noi la thu do cua Viet Nam"
>>> print(cau)
Ha Noi la thu do cua Viet Nam
```

a. Các phương thức

- Tạo chuỗi con: sử dụng [index] hoặc [from:to], hoặc [form]
Với index(chỉ mục) bắt đầu từ 0, đến chiều dài chuỗi -1

Ví dụ 6.19.

```
>>> greeting = "Hello Python"
>>> print(greeting)
Hello Python
>>>
>>> print(greeting[0])
H
>>>
>>> print(greeting[2:5])
llo
>>>
>>> print(greeting[3:])
lo Python
```

Ví dụ 6.20. `>>> greeting="Hello Python"`

`>>> print(greeting[-5:-2])`

yth

Kết quả trả về một chuỗi con từ vị trí 3 đến 5 từ cuối chuỗi đã cho

Nối chuỗi: sử dụng toán tử +

Ví dụ 6.21 `>>> greeting="Hello"`

`>>> name="Python"`

`>>> print(greeting+name)`

HelloPython

*Lặp chuỗi: sử dụng toán tử **

Ví dụ 6.22 `>>> greeting="Hello Python"`

`>>> print(greeting*4)`

Hello PythonHello PythonHello PythonHello Python

Lấy chiều dài của chuỗi: sử dụng phương thức len()

Ví dụ 6.23 `>>> greeting="Hello Python"`

`>>> print(len(greeting))`

Ký tự	Ý nghĩa	Ví dụ
'd'	Số nguyên có dấu	<pre>print('%d' % (-100)) >>> -100</pre>
'i'	Số nguyên có dấu	<pre>print('%i' % (-100)) >>> -100</pre>
'o'	Giá trị dạng bát phân (octan)	<pre>print('%o' % (20)) >>> 24</pre>
'u'	Giống với ký tự 'd'	<pre>print('%u' % (-100)) >>> -100</pre>
'x'	Giá trị hệ 16 có dấu (viết thường)	<pre>print("%5x" % (47)) >>> 2f</pre>
'X'	Giá trị hệ 16 có dấu (viết hoa)	<pre>print("%5.4X" % (47)) >>> 002F</pre>
'e'	Định dạng số mũ cho số thực dấu chấm động (viết thường)	<pre>print("%9.2e" % (312.087)) >>> 3.12e+02</pre>
'E'	Định dạng số mũ cho số thực dấu chấm động (viết thường)	<pre>print("%9.2E" % (312.087)) >>> 3.12E+02</pre>
'f'	Định dạng số thực dạng thập phân	<pre>print('%f' % (100.21)) >>> 100.210000</pre>
'F'	Định dạng số thực dạng thập phân	<pre>print('%F' % (100.21)) >>> 100.210000</pre>
'g'	Định dạng số thực dấu chấm động. Sử dụng định dạng số mũ viết thường nếu số mũ nhỏ hơn -4 hoặc nhỏ hơn độ chính xác	<pre>print('%g' % (3e9)) >>> 3e+09</pre>
'G'	Định dạng số thực dấu chấm động. Sử dụng định dạng số mũ viết hoa nếu số mũ nhỏ hơn -4 hoặc nhỏ hơn độ chính xác	<pre>print('%G' % (3e9)) >>> 3E+09</pre>
'c'	Một ký tự (nhận vào một giá trị số nguyên hoặc một ký tự)	<pre>print('%c' % ('a')) >>> a</pre>
'r'	Chuỗi ký tự (chuyển đổi từ bất kỳ đối tượng Python nào bằng hàm repr())	<pre>print('%r' % ('Python')) >>> 'Python'</pre>
's'	Chuỗi ký tự (chuyển đổi từ bất kỳ đối tượng Python nào bằng hàm str())	<pre>print('%s' % ('Python')) >>> Python</pre>
'a'	Chuỗi ký tự (chuyển đổi từ bất kỳ đối tượng Python nào bằng hàm ascii())	<pre>print('%a' % ('Python cơ bản')) >>> 'Python c\u01a1 b\u1ea3n'</pre>
'%'	Không có tham số nào được chuyển đổi, trả về kết quả là ký tự dấu phần trăm '%'	<pre>print('%') >>> %</pre>

b. Định dạng chuỗi

Bảng 6.1. Định dạng chuỗi ký tự

Ký tự	Ý nghĩa	Ví dụ
'd'	Số nguyên có dấu	print('%d' %(-100)) >>> -100
'i'	Số nguyên có dấu	print('%i' %(-100)) >>> -100
'o'	Giá trị dạng bát phân (octan)	print('%o' %(20)) >>> 24
'u'	Giống với ký tự 'd'	print('%u' %(-100)) >>> -100
'x'	Giá trị hệ 16 có dấu (viết thường)	print("%5x"% (47)) >>> 2f
'X'	Giá trị hệ 16 có dấu (viết hoa)	print("%5.4X"% (47)) >>> 002F
'e'	Định dạng số mũ cho số thực dấu chấm động (viết thường)	print("%9.2e"% (312.087)) >>> 3.12e+02

Bảng 6.1. Định dạng chuỗi ký tự

‘E’	Định dạng số mũ cho số thực dấu chấm động (viết thường)	<pre>print("%9.2E"% (312.087)) >>> 3.12E+02</pre>
‘f’	Định dạng số thực dạng thập phân	<pre>print('%f' %(100.21)) >>> 100.210000</pre>
‘F’	Định dạng số thực dạng thập phân	<pre>print('%F' %(100.21)) >>> 100.210000</pre>
‘g’	Định dạng số thực dấu chấm động. Sử dụng định dạng số mũ viết thường nếu số mũ nhỏ hơn -4 hoặc nhỏ hơn độ chính xác	<pre>print('%g' %(3e9)) >>> 3e+09</pre>
‘G’	Định dạng số thực dấu chấm động. Sử dụng định dạng số mũ viết hoa nếu số mũ nhỏ hơn -4 hoặc nhỏ hơn độ chính xác	<pre>print('%G' %(3e9)) >>> 3E+09</pre>
‘c’	Một ký tự (nhận vào một giá trị số nguyên hoặc một ký tự)	<pre>print('%c' %('a')) >>> a</pre>
‘r’	Chuỗi ký tự (chuyển đổi từ bất kỳ đối tượng Python nào bằng hàm repr())	<pre>print('%r' %('Python')) >>> 'Python'</pre>

Bảng 6.1. Định dạng chuỗi ký tự

's'	Chuỗi ký tự (chuyển đổi từ bất kỳ đối tượng Python nào bằng hàm str())	print('%s' %('Python')) >>> Python
'a'	Chuỗi ký tự (chuyển đổi từ bất kỳ đối tượng Python nào bằng hàm ascii())	print('%a' %('Python cơ bản')) >>> 'Python c\u01a1 b\u1ea3n'
'%'	Không có tham số nào được chuyển đổi, trả về kết quả là ký tự dấu phần trăm '%'	print('%') >>> %

Ví dụ 6.24.

```
>>> name = "Jonny"
>>> age = 25
>>> height = 1.60
>>> weight = 52.5
>>> print("My name is %s. I'm %i years old. My height is %.2f(m) and weight is %.2f(kg)" % (name, age, height, weight))
My name is Jonny. I'm 25 years old. My height is 1.60(m) and weight is 52.50(kg)
```

Định dạng các biến theo kiểu dữ liệu tương ứng ký tự định dạng 1, 2, 3, 4

c. Định dạng sử dụng phương thức format()

- ❑ Phương thức format() có nhiệm vụ thực hiện định dạng (các) giá trị được chỉ định và chèn các giá trị này vào bên trong **đối tượng giữ chỗ** của chuỗi ký tự.
- ❑ Đối tượng giữ chỗ được xác định bằng dấu ngoặc nhọn: {}.

Cú pháp

```
string.format(value1, value2...)
```

Trong đó:

- Tham số value1 và value2 là bắt buộc. Một hoặc nhiều giá trị cần được định dạng và chèn vào chuỗi ký tự.
- Các giá trị là danh sách các giá trị được phân tách bằng dấu phẩy.
- Các giá trị có thể thuộc bất kỳ kiểu dữ liệu nào.

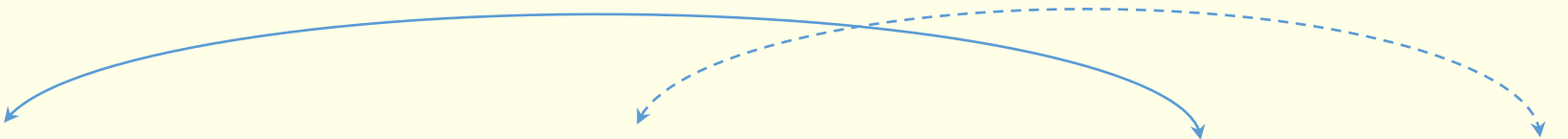
c. Định dạng sử dụng phương thức format(),...

Ví dụ 6.25.

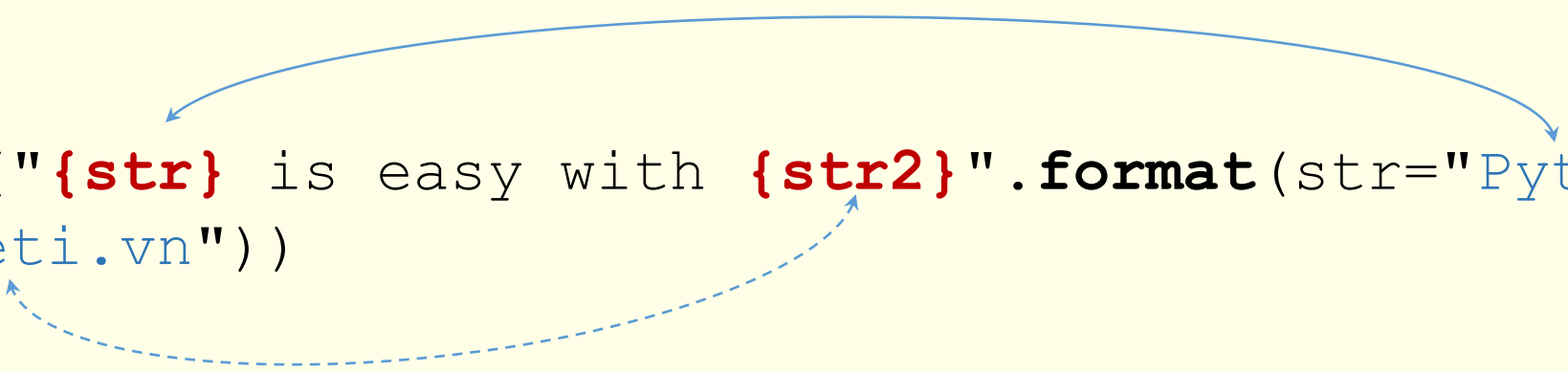
```
>>> print("My name is {}, you can learn about me,".format("Python"))
```



```
>>> print("{0} is easy with {1}".format("Python", "uneti.vn"))
```



```
>>> print("{str} is easy with {str2}".format(str="Python",  
str2="Uneti.vn"))
```



Kết quả output:

■ My name is Python, you can learn about me,

■ Python is easy with uneti.vn

■ Python is easy with Uneti.vn

in ra chuỗi ký tự với các đối tượng giữ chỗ tương ứng (Python).

đối tượng giữ chỗ với giá trị số nguyên được truyền vào

các đối tượng giữ chỗ với các từ khóa

c. Định dạng sử dụng phương thức format(),...

- ❑ Để định dạng các chuỗi ký tự một cách mềm dẻo hơn, chúng ta thêm các chỉ thị định dạng vào trong dấu ngoặc nhọn {}.

Ví dụ 6.26.

```
>>> print("Number {:d}".format(1982))
Number 1982
>>> print("Float number {:f}".format(395.19999))
Float number 395.199990
>>> print("Binary {0:b}, Oct:{0:o}, Hex: {0:x}".format(19))
Binary 10011, Oct:23, Hex: 13
```

Trong đoạn mã ở ví dụ 6.26 trên,

- Câu lệnh print đầu tiên sẽ thực hiện in ra định dạng số nguyên (với chỉ thị **:d**),
- Câu lệnh print thứ hai in ra giá trị số thực dạng thập phân (chỉ thị **:f**).
- Câu lệnh print thứ ba thực hiện in ra các định dạng cơ số 2 (chỉ thị **:b**), 8 (chỉ thị **:o**) và 16 (chỉ thị **:x**) của giá trị 19.

Bảng 6.2. hiển thị dữ liệu ra màn hình khi sử dụng hàm print()

Kiểu định dạng	Mô tả
:<	Căn lề bên trái với một lượng khoảng trống nhất định
:>	Căn lề bên phải với một lượng khoảng trống nhất định
:^	Căn lề chính giữa với một lượng khoảng trống nhất định
:=	Đặt dấu ở vị trí bên trái xa nhất
:+	Chỉ định kết quả là số dương hay số âm
:−	Chỉ định kết quả là số âm
:	Sử dụng một khoảng trắng để thêm khoảng trắng trước các giá trị số dương
:,	Sử dụng dấu phẩy làm dấu phân tách hàng nghìn
:_	Sử dụng dấu gạch dưới làm dấu phân tách hàng nghìn
:b	Định dạng số nhị phân
:c	Chuyển đổi giá trị sang các bộ ký tự unicode tương ứng
:d	Định dạng số thập phân
:e	Định dạng số học, với ký tự e viết thường
:f	Cố định định dạng số thập phân
:g	Định dạng chung
:o	Định dạng số bát phân
:x	Định dạng số hệ 16

6.4. CHUYỂN ĐỔI KIỂU DỮ LIỆU

6.4.1. Mục tiêu

Xác định kiểu dữ liệu: sử dụng hàm **type()**

Ví dụ 6.27.

```
>>> x="Hello"  
>>> type(x)  
<class 'str'>  
>>> x=3.5  
>>> type(x)  
<class 'float'>  
>>> x=3  
>>> type(x)  
<class 'int'>  
>>> x=5>4  
>>> type(x)  
<class 'bool'>
```

6.4.2. Các phương thức chuyển đổi kiểu dữ liệu

Bảng 6.3. Một số phương thức chuyển đổi kiểu dữ liệu

Phương thức	Mô tả
int (x)	Chuyển x thành integer, với x có kiểu chuỗi.
float (x)	Converts x thành floating-point number.
complex (real [,imag])	Tạo một complex number.
str (x)	Chuyển đối tượng x thành chuỗi.
repr (x)	Chuyển đối tượng x thành một chuỗi - expression string.
eval (str)	Đánh giá một chuỗi và trả về một object.
chr (x)	Chuyển integer x thành một ký tự.
ord (x)	Chuyển ký tự x thành giá trị integer của nó.
hex (x)	Chuyển một integer x thành chuỗi hexadecimal.
oct (x)	Chuyển một integer x thành chuỗi octal.

Ví dụ 6.28.

```
strInt="12"  
print(int(strInt)*2)  
r1=25  
em=3  
complexNum=complex(r1,em)  
print(complexNum)  
numEval=eval(strInt)  
print(numEval+numEval)
```



```
24  
(25+3j)  
24
```

Ví dụ 6.29. Trong Python shell gõ các lệnh sau

```
>>> chr(98)  
'b'  
>>> ord('c')  
99  
>>> hex(100)  
'0x64'  
>>> oct(100)  
'0o144'
```

2.4.3. Chú thích trong Python

- Chú thích (comment) là những dòng ghi chú, giải thích cho source code trong chương trình.
- Phần chú thích có thể ghi thông tin tác giả, ngày viết, version hoặc giải thích cho một đoạn chương trình khó...
- Khi chạy chương trình, trình biên dịch/thông dịch sẽ không biên dịch/thông dịch phần chú thích này.

Cú pháp

- Khi chú thích trong một dòng sử dụng ký tự '#'
- Khi đoạn chú thích nằm trên nhiều dòng sử dụng 3 ký tự nháy kép "'''" để bắt đầu đoạn chú thích và ba dấu nháy kép "'''" để đóng nội dung đoạn chú thích.

Ví dụ 6.30.

```
1  #Ví dụ chuyển đổi kiểu dữ liệu
2  strInt="12"
3  print(int(strInt)*2)    #ép kiểu strInt từ chuỗi --> int
4  r1=25
5  em=3
6  complexNum=complex(r1,em)
7  print(complexNum)
8  numEval=eval(strInt)   #ép kiểu strInt sang dạng số và gán cho biến numval
9  print(numEval+numEval)
```



```
24
(25+3j)
24
```

6.5. NHẬP XUẤT DỮ LIỆU TRÊN SHELL

Ứng dụng trên shell (console)

- Là ứng dụng nhập xuất ở chế độ văn bản tương tự như màn hình Console của hệ điều hành MS-DOS.
- Các ứng dụng kiểu shell thường được dùng để minh họa các ví dụ cơ bản liên quan đến cú pháp ngôn ngữ, các thuật toán, và các chương trình ứng dụng không cần thiết đến giao diện người dùng đồ họa.

6.5.1. Nhập dữ liệu

Ví dụ 6.31.

```
>>> name = input("What's your name?")
What's your name? Ng Van Minh
>>> print("My name is ", name)
My name is   Ng Van Minh
```

Ví dụ 6.32. (lỗi)

```
>>> toan = input("Diem mon toan: ")
Diem mon toan: 8.0
>>> van = input("Diem mon van: ")
Diem mon van: 5.5
>>> trung_binh = (toan+van)/2
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    trung_binh = (toan+van)/2
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

Ví dụ 6.33. (chuyển đổi kiểu)

```
>>> toan = float(input("Diem mon toan: "))
Diem mon toan: 8.0
>>> van = float(input("Diem mon van: "))
Diem mon van: 5.5
>>> trung_binh = (toan+van)/2
>>> print("Diem trung binh Toan Van la: ", trung_binh)
Diem trung binh Toan Van la:  6.75
```

- Sử dụng phương thức kết hợp: **eval(input(prompt))**

Ví dụ 6.34.

```
>>> toan = eval(input("Diem mon toan: "))
Diem mon toan: 8.0
>>> van = eval(input("Diem mon van: "))
Diem mon van: 5.5
>>> trung_binh = (toan+van)/2
>>> print("Diem trung binh Toan Van la: ", trung_binh)
Diem trung binh Toan Van la:  6.75
```

```
>>> age = eval(input("How old are you? "))
How old are you? 20
>>> print("I'm %i years old. " %age)
I'm 20 years old.
```

6.5.2. Xuất dữ liệu

❑ Python cung cấp phương thức `print(...)` để xuất dữ liệu

Dạng 1

Ví dụ 6.36

```
1  #print dạng 1
2  print("Hello" + " the World. ")
3  name = "Alice"
4  print("Hello "+ name + ".")
5  age = 21
6  print("My name is "+ name+". I am " + str(age)+ " years old.")
```

```
Hello the World.
Hello Alice.
My name is Alice. I am 21 years old.
```

Dạng 2

Ví dụ 6.37

```
1  #print dạng 2
2  print("\n")
3  print("Hello" + " the World. ")
4  name = "Alice"
5  print("Hello "+ name + ".")
6  age = 21
7  print("My name is "+ name + ". I am " , age , " years old.")
```

Dạng 3

Ví dụ 6.38

```
1  #print dạng 3
2  print("\n")
3  print("Hello" + " the World. ")
4  name = "Alice"
5  print("Hello "+ name + ".")
6  age = 21
7  print("My name is  %s  I am %d  years old." %(name,age))
```

Kết quả thực thi các ví dụ 6.37, 6.38

```
Hello the World.
Hello Alice.
My name is  Alice  I am 21  years old.
```


Ví dụ 6.39.

Viết chương trình cho phép người dùng nhập vào 2 số, tính và in ra tổng/hiệu/tích/thương của 2 số đã nhập

```
1  x=eval(input("Nhập x : "))
2  y=eval(input("Nhập y : "))
3  tong = x + y
4  hieu = x - y
5  tich = x*y
6  thuong=x/y
7  print("Tổng = %d, hiệu =%d, tích=%d, thương = %0.1f"%(tong, hieu, tich, thuong))
```

Kết quả:

```
Nhập x : 5
Nhập y : 10
Tổng = 15, hiệu =-5, tích=50, thương =_0.5
```

Câu hỏi thảo luận

1. Anh/Chị hãy giải thích tại sao một biến nhớ trong Python tại các thời điểm khác nhau có thể gán các giá trị dữ liệu có kiểu khác nhau?
2. Nêu Cách khai báo và sử dụng biến? cho ví dụ?
3. Anh/Chị cho biết có thể khai báo lại một biến không?
4. Có thể khai báo và sử dụng kết hợp các loại dữ liệu khác nhau như chuỗi ký tự và số được không? Ví dụ?
5. Nêu phương thức xóa một biến?

Bài tập vận dụng

Các bài tập trong TLHT chương 6.