Softech®
Gather to thrive together

Aptech™
COMPUTER EDUCATION
Unleash your potential

BÀI 2

Biến và Kiểu dữ liệu

Tóm Tắt Nội Dung

Trong bài học này chúng ta sẽ đi tìm hiểu lần lượt các nội dung

1. Cách khai báo BIẾN trong Python
2. Nguyên tắc đặt tên BIẾN
3. Các kiểu dữ liệu của BIẾN
4. Dữ liệu kiểu số và Các toán tử số học trong Python
5. Dữ liệu kiểu chuỗi và các phương thức xử lý chuỗi

2.1 Cách khai báo BIẾN trong Python

Định nghĩa: Biến là gì ?

Biến (variable) là một khái niệm cơ bản trong lập trình, nó là tên của một khu vực trong bộ nhớ của máy tính, được sử dụng để lưu trữ và đại diện cho các giá trị dữ liệu như số, chuỗi, đối tượng, và nhiều loại dữ liệu khác

Khi ta tạo một biến trong Python, ta đang cung cấp một tên duy nhất cho biến đó và gán cho nó một giá trị cụ thể. Biến có thể được sử dụng để tham chiếu đến giá trị của nó trong các phép toán, điều kiện, hoặc các phần khác của chương trình.

Biến có thể chứa dữ liệu kiểu số (int), chuỗi (str), logic (bool)…

2.1 Cách khai báo BIẾN trong Python

Cú pháp khai báo biến

tên_biến = giá_trị

- **tên_biến** là tên mà ta muốn đặt cho biến
- **giá_trị** là giá trị mà ta muốn gán cho biến

Khi khai báo biến trong Python ta không cần phải khai báo kiểu dữ liệu ta chỉ cần gán một giá trị cho tên biến, Python sẽ tự động suy ra kiểu dữ liệu tương cho biến đó.

Sử dụng hàm **type()** để kiểm tra kiểu dữ liệu hiện tại của biến

2.1 Cách khai báo BIẾN trong Python

Ví dụ về khai báo biến

Ví dụ để định nghĩa biến **name** với giá trị **Python** ta viết như sau:

```
name  = ‘Python’ # name là kiểu chuỗi str
```

Ví dụ để định nghĩa biến **age** với giá trị **35** ta viết như sau:

```
age  = 35 # age là kiểu số int
```

Kiểm tra kiểu dữ liệu hiện tại của biến

```
print(type(name)) # Kết quả in ra Terminal: <type 'str'>
```

2.2 Nguyên tắc đặt tên biến trong Python

1. Tên biến chỉ chứa các ký tự a-z, A-Z, 0-9 và dấu gạch dưới _
2. Phải bắt đầu bằng một ký tự chữ a-z, A-Z hoặc dấu gạch dưới _
3. Tên biến không bắt đầu bằng một con số
4. Tên biến phân biệt chữ hoa chữ thường (age, Age , AGE là 3 biến khác nhau)
5. Tên biến không chứa các từ khóa của Python

Langue_name = ‘Python’

Từ khóa trong Python

Keywords

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

Soft Keywords

```
match      case       _
```

2.2 Nguyên tắc đặt tên biến trong Python

Một số cách đặt **hợp lệ**

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

Một số cách đặt **không hợp lệ**

```
2myvar = "John"
my-var = "John"
my var = "John"
```

Đặt tên kiểu **Snake Case**

```
my_variable_name  = "John"
```

Đặt tên kiểu **Pascal Case**

```
MyVariableName  = "John"
```

Đặt tên kiểu **Camel Case**

```
myVariableName  = "John"
```

2.2 Nguyên tắc đặt tên biến trong Python

Gán nhiều giá trị cho nhiều biến

```
x, y, z  =  "Orange", "Banana", "Cherry"
print(x) # x = Orange
print(y) # y = Banana
print(z) # z = Cherry
```

Hoán đổi giá trị của biến gán đồng thời

```
x, y   =  1, 2
print(x, y) # (1, 2)
x, y = y, x  # Hoán đổi giá trị x và y cho nhau
print(x, y) # (2, 1)
```

2.2 Nguyên tắc đặt tên biến trong Python

Gán một giá trị cho nhiều biến

```
x = y = z  =  "Orange“
print(x) # x = Orange
print(y) # y = Orange
print(z) # z = Orange
```

Xuất giá trị của biến (Output variables)

```
x =  “Hello“
y =  “Python“
print(x, y)
print(x + y)
```

2.4 Các Kiểu dữ liệu trong Python

Ví dụ	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	Range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = None	NoneType

2.4 Các Kiểu dữ liệu trong Python

Ép kiểu dữ liệu	Data Type
x = str("Hello World")	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list(("apple", "banana", "cherry"))	list
x = tuple(("apple", "banana", "cherry"))	tuple
x = range(6)	Range
x = dict(name="John", age=36)	dict
x = set(("apple", "banana", "cherry"))	set
x = frozenset(("apple", "banana", "cherry"))	frozenset
x = bool(5)	bool

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.1 Số trong Python - Python Numbers

Có 3 loại số trong Python: int, float, complex

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

```
print(type(x))
```

Dùng hàm type() để kiểm tra kiểu dữ liệu của biến

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.1 Số trong Python - Python Numbers

int

Hay integer, bao gồm số, số nguyên âm, số nguyên dương, không bao gồm số thập phân, không giới hạn độ dài

```
x = 1
y = 43545454656
z =  -8547584
```

float

là một số, dương hoặc âm, chứa một hoặc nhiều số thập phân.

```
x = 1.1
y = 1.02
z =  -8,78
```

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

Phép tính	Ý nghĩa	Ví dụ
+	Cộng	x + y
-	Trừ	x - y
*	Nhân	x * y
/	Chia	x / y
%	Chia lấy phần dư	x % y
**	Lũy thừa	x ** y
//	Chia lấy phần nguyên	x // y

Trong Python, phép tính được thực hiện theo thứ tự ưu tiên sau đây:

1. Các phép tính trong dấu ngoặc đầu tiên được thực hiện trước.
2. Các phép nhân và chia được thực hiện trước các phép cộng và trừ.
3. Các phép cộng và trừ được thực hiện từ trái sang phải.

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

Thứ tự ưu tiên được mô tả trong bảng bên dưới, bắt đầu với ưu tiên cao nhất ở trên cùng:

Operator	Description
()	Trong ngoặc
**	Lũy thừa
+x -x ~x	Cộng một ngôi, trừ một ngôi và theo chiều bit KHÔNG
* / // %	Phép nhân, phép chia, phép chia sàn và lấy dư (mod)
+ -	Phép cộng, trừ
<< >>	Dịch chuyển trái và phải theo từng bit

Xem tiếp trang sau

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

Thứ tự ưu tiên các phép tính trong Python

Operator	Description
^	Bitwise XOR
|	Bitwise OR
== != > >= < <= is is not in not in	Toán tử so sánh: bằng, không bằng, lớn hơn, lớn hơn hoặc bằng, nhỏ hơn, nhỏ hơn hoặc bằng
not	Logical NOT
and	AND
or	OR

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

Một số ví dụ về toán tử số học

```
x = 5
y = 3
print(x + y) # Phép cộng
print(x - y) # Phép trừ
print(x * y) # Phép nhân
print(x / y) # Phép chia
print(x % y) # Chia Mod
print(x // y) # Chia sàn
```

Biểu thức nhiều phép tính

```
print(5 + 4 - 7 + 3)
```

Tình từ trái qua phải
5 + 4 = 9
9 - 7 = 2
2 + 3 = 5 là kết quả

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

```
result = 2 + 3 * 4 - 6 / 2 ** 2
```

Trong ví dụ này, chúng ta có một biểu thức gồm các toán tử số học: cộng (+), nhân (*), chia (/), và lũy thừa (**). Ta sẽ giải thích từng bước của phép tính theo thứ tự ưu tiên

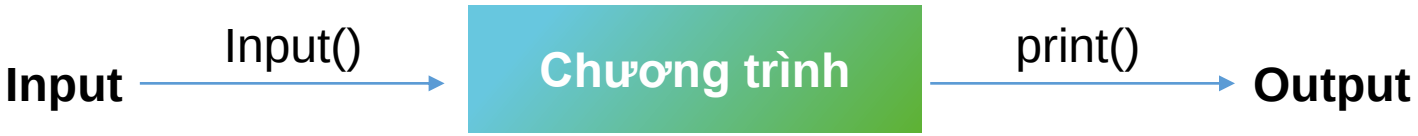
- 2 ** 2: Thực hiện phép lũy thừa. 2 ** 2 tương đương với 2^2 = 4
- 3 * 4: Thực hiện phép nhân. Kết quả của phép nhân là 12
- 6 / 4: Thực hiện phép chia. Kết quả của phép chia là 1.5
- 2 + 12: Thực hiện phép cộng. Kết quả của phép cộng là 14
- 14 - 1.5: Thực hiện phép trừ.
- Kết quả của phép trừ là 12.5

2.7 Tìm hiểu về lệnh input()

Định nghĩa: Input là gì ?

Trong mọi chương trình hệ thống đều có sự tương tác dữ liệu 2 chiều: đầu vào (input) và đầu ra (output)

Trong các bài học trước thì chúng ta đã làm quen với lệnh **print** đóng vai trò là đâu ra của chương trình. Bài này chúng ta sẽ làm quen với input là đầu vào



Hàm **input()** dùng thực hiện việc **lấy dữ liệu nhập từ bàn phím** làm giá trị đầu vào cho chương trình

2.7 Tìm hiểu về lệnh input()

Cú pháp lệnh input()

tên_biến = input(‘Nội dung gợi ý’)

- **tên_biến** là tên mà ta muốn đặt cho biến
- **Nội dung gợi ý** là là câu từ mà mình hiển thị ra để người dùng biết nên nhập cái gì vào từ bàn phím

Ví dụ:

```
name  = input(‘Vui lòng nhập vào tên của bạn: ’)
print(‘Xin chào’ + name)
```

2.7 Tìm hiểu về lệnh input()

Ví dụ về toán tử với input()

Lấy giá trị 2 biến x và y nhập lên từ bàn phím là thực hiện các phép tính với chúng”

```
x  = input(‘Vui lòng nhập giá trị của biến x: ’) #Ví dụ nhập vào: 5
y  = input(‘Vui lòng nhập giá trị của biến y: ’) #Ví dụ nhập vào: 3

print(x + y) # Kết quả in ra Terminal: 53
```

Tại sao kết quả không phải là 8 mà là 53 ?

Giá trị nhận được từ input() luôn là một chuỗi

Do vậy với toán tử + trên nó trở thành lệnh nối chuỗi chứ không thực hiện một phép tính

2.7 Tìm hiểu về lệnh input()

Chuyển đổi dữ liệu

Để có được kiểu dữ liệu mình muốn, chúng ta có thể sử dụng các hàm int(), float(), str() để chuyển đổi kiểu dữ liệu:

Int() **Giá trị trả về = int(<giá trị>)**

```
x = '2‘ # x đang là str có giá trị = 2
print(int(x)) # x giờ là int = 2
```

float() **Giá trị trả về = int(<giá trị>)**

```
x = ‘2.74156‘ # x đang là str có giá trị = ‘2.74156’
print(float(x)) # x giờ là int = 2.74156
```

strt() Thì ngược lại, chuyển dữ liệu từ kiểu số sang kiểu chuỗi

2.7 Tìm hiểu về lệnh input()

Eval() tính toán giá trị từ một biểu thức trong chuỗi

Cú pháp **Giá trị trả về = eval("<biểu thức>")**

```
print(eval("12 + 7"))
# kết quả cho ra một số : 19
```

Tính toán giá trị từ biểu thức nhập vào bằng input()

```
x = eval(input("Nhập vào biểu thức toán học: "))
print("Giá trị của biểu thức x là", x)
```

Nhận biết đồng thời nhiều biểu thức trên một hang, cách bởi dấu ,

```
m, n, p = eval(input("Nhập vào các số m, n, p (cách nhau bởi dấu phẩy) "))
print("Các số đã nhập là", m,n, p)
```

2.8 Tổng quan về Chuỗi (String)

Giới thiệu về Kiểu dữ liệu String

- Trong Python, kiểu dữ liệu chuỗi (string) là một dạng dữ liệu dùng để lưu trữ văn bản, chữ số, ký tự, hoặc bất kỳ dãy các ký tự nào

- Chuỗi trong Python được biểu diễn bằng các ký tự nằm trong cặp dấu nháy đơn ('...') hoặc nháy kép ("..."). Python hỗ trợ các thao tác và phương thức mạnh mẽ để làm việc với kiểu dữ liệu chuỗi

```
# Khai báo chuỗi bằng cặp dấu nháy đơn
string1 = 'Hello, World!'
# Khai báo chuỗi bằng cặp dấu nháy kép
string2 = "Python is awesome"
# Chuỗi có thể chứa các ký tự đặc biệt
string3 = 'Các ký tự đặc biệt: !@#$%^&*()'
# Chuỗi có thể chứa các số và chữ số
string4 = '12345'
```

2.8 Tổng quan về Chuỗi (String)

- Chuỗi nhiều dòng với 3 kí tự nháy kép """

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

- Hoặc Chuỗi nhiều dòng với 3 kí tự nháy đơn '''

```
a =  '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

2.8 Tổng quan về Chuỗi (String)

- Trong Python String cũng được coi là một Mảng (Array) tập hợp của nhiều kí tự, vị trí sắp xếp bắt đầu là số 0

```
a = "Hello, World!"
print(a[1]) #output: e
```

- Vì là mảng nên bạn có thể lặp với vòng lặp **for**

```
for x in "banana":
  print(x)
```

- Dùng phương thức len() để kiểm tra độ dài của chuỗi có bao nhiêu kí tự

```
a = "Hello, World!"
print(len(a)) #output: 13
```

2.8 Tổng quan về Chuỗi (String)

- Kiểm tra sự tồn tại của một kí tự trong chuỗi với từ khóa **in**

```
txt = "The best things in life are free!"
print("free" in txt)

if "free" in txt:
  print("Có tồn tại")
```

- Hoặc kết hợp với **not** để kiểm tra không tồn tại

```
if "fresh" not in txt:
    print("Đúng, fresh không tồn tại trong chuỗi")
```

2.9 Các phương thức xử lí Chuỗi

Cắt chuỗi

Bạn có thể tách chuỗi dự vào vị trí index của nó trong chuỗi

Cú pháp: **string[start:end]**

- start mặc định là 0, có thể là số nguyên âm
- end mặc định là đến cuối chuỗi, có thể là số nguyên âm

```
b = "Hello, World!"
print(b[2:5]) #output: llo
```

Ví dụ này sẽ lấy kí tự từ vị trí thứ 2 đến 4 (5 – 1, không bao gồm vị trí số 5)

```
print(b[-2:-5]) #output: orl
```

Dùng số nguyên âm để đảo chiều vị trí lấy

2.9 Các phương thức xử lí Chuỗi

Cắt chuỗi

Ví dụ không điền **start**, thì start mặc định là số 0

```
b = "Hello, World!"
print(b[0:5]) #output: Hello
```

Ví dụ không điền **end**, thì start mặc định là đến vị trí cuối

```
print(b[2:]) #output: llo, World!
```

2.9 Các phương thức xử lí Chuỗi

Biến đổi chuỗi

- Biến thành chuỗi viết HOA

```
a = "Hello, World!"
print(a.upper()) #output: HELLO, WORLD!
```

- Biến thành chuỗi viết thường

```
print(a.lower()) #output: hello, world!
```

- Kí tự đầu tiên mỗi chữ viết HOA

```
txt = "Welcome to my world"
print(txt.title()) #output: Welcome To My World
```

2.9 Các phương thức xử lí Chuỗi

Biến đổi chuỗi

- Tách chuỗi với **split()** dựa vào một kí tự nằm giữa chuỗi

```
a = "Hello, World!"
print(a.split(",")) #returns list:  ['Hello', ' World!']
```

- Xóa khoảng trắng 2 đầu chuỗi với **strip()**

```
a = " Hello, World! "
print(a.strip())  # returns "Hello, World!"
```

- Thay thế kí tự trong chuỗi với **replace()**

```
txt = "Welcome to my world"
print(txt.replace("world", "home")) #output: Welcome To My home
```

2.9 Định dạng Chuỗi

Định dạng (đóng khuôn) chuỗi

- Trong python, không thể nối chuỗi với số với toán tử + cộng, nhưng chúng ta có thể làm điều đó với hàm **format()**

```
age = 36
txt = “Tôi là  John, Tôi {} tuổi"
print(txt.format(age))
#Kết quả in ra: Tôi là  John, Tôi 36 tuổi
```

Trong ví dụ trên kí tự {} đóng vai trò giữ chỗ cho biến age, và age sẽ hiển thị ra thay cho {}. Các vị trí {} này được gọi là **trường** (hay là field)

2.9 Định dạng Chuỗi

Định dạng (đóng khuôn) chuỗi

- **Thay thế đơn giản không chỉ số**

```
age = 36
name = ‘Jonh’
txt = “Tôi là  {}, Tôi {} tuổi"
print(txt.format(name, age))
#Kết quả in ra: Tôi là  John, Tôi 36 tuổi
```

- Khi đó biến **name** sẽ thay thế vào kí tự **{}** đầu tiên trong chuỗi txt
- Và biến **age** sẽ thay thế vào vị trí **{}** thứ 2 trong chuỗi txt
- Nếu có một biến khác thì biến và {} được tham chiếu một cách tuần tự như vậy khi bạn thêm vào hàm **format()**

Lưu ý: bạn có thể đưa nhiều tham số biến hơn số lượng {} cần thay thế

2.9 Định dạng Chuỗi

Định dạng (đóng khuôn) chuỗi

- **Thay thế có chỉ số**

```
quantity  = 2
price  = 49.95
txt = “Mua {0} cái bánh hết {1} đô la. Em tôi ăn hết {0} cái bánh"
print(txt.format(quantity, price))
#Kết quả in ra: Mua 2 cái bánh hết 49.95 đô la. Em tôi ăn hết 2 cái bánh
```

Với cách dùng chỉ số này thì nhìn vào dễ hiểu hơn:
- **quanlity:** sẽ tương ứng với field số {0}
- **price:** sẽ tương ứng với field số {1}

Và mình có thể đặt các field bất kỳ ở đâu chỉ cần ghi số vào là được.

Lưu ý: nếu đã đánh số thì tất cả các field đều đánh số, chứ không được cái thì {}, cái thì {0}. Python sẽ không xử lý đúng.

2.9 Định dạng Chuỗi

- **Tạo khuôn bằng f<string>**

s = f"{<var1> : <Chỉ lệnh>} {<var2> : <Chỉ lệnh>}"

Đây là một cách đơn giản hơn để có thể trộn biến vào trong chuỗi

```
mark = 8
name = 'Toán'
txt = "Môn {} được {} điểm"
print(txt.format(name, age))
#Kết quả in ra:
Môn Toán được 8 điểm
```

```
mark = 8
name = 'Toán'
txt = f"Môn {name} được {mark:*>5.2f} điểm"
print(txt)
#Kết quả in ra:
Môn Toán được *8.00 điểm
```

Cách không tường minh

Chỉ lệnh vẫn được áp dụng với f<string>

Giúp code trở nên gọn hơn

3 Escape Character

Giới thiệu về khái niệm Escape

- Trong Python, ký tự escape (escape character) là một ký tự đặc biệt được sử dụng để thể hiện các ký tự không thể hiện hoặc có ý nghĩa đặc biệt trong chuỗi. Khi bạn sử dụng ký tự escape, nó cho phép bạn đưa vào chuỗi những ký tự đặc biệt như dấu nháy kép, dấu nháy đơn, dấu gạch chéo ngược, và các ký tự khác mà không bị xem là kết thúc của chuỗi hoặc gây ra sự hiểu lầm về cú pháp.

  Ký tự escape được ký hiệu bằng một dấu gạch chéo ngược ("") theo sau là ký tự đại diện cho ý nghĩa cụ thể. Dưới đây là một số ví dụ về ký tự escape phổ biến trong Python:

  \n: Ký tự xuống dòng (newline)
  \t: Ký tự tab ngang (tab)
  \": Dấu nháy kép
  \': Dấu nháy đơn

3 Escape Character

Ví dụ về cách sử dụng Escape

Sử dụng ký tự escape để in ra các ký tự đặc biệt trong chuỗi

```
print("Dòng 1\nDòng 2")
# In ra hai dòng cách nhau bởi dấu xuống dòng (\n)
print("Tab:\t\tKý tự tiếp theo sau tab")
# In ra ký tự tab (\t) để tạo khoảng cách
print("Dấu nháy kép: \"Ví dụ\"  ")
# In ra dấu nháy kép (") trong chuỗi
print('Dấu nháy đơn: I\'m a student ')
# In ra dấu nháy đơn (') trong chuỗi
print("Dấu gạch chéo ngược: \\")
# In ra dấu gạch chéo ngược (\) trong chuỗi
```

Tham khảo link sau để tìm hiểu tất cả phương thức xử lý chuỗi:
https://www.w3schools.com/python/python_strings_methods.asp

3.1 Dữ liệu kiểu Số và toán tử số học trong Python

Nguồn tham chiếu

- Biến trong Python
  https://www.w3schools.com/python/python_variables.asp
- Kiểu dữ liệu của biến
  https://www.w3schools.com/python/python_datatypes.asp
- Toán tử trong Python
  https://www.w3schools.com/python/python_operators.asp
- Chuỗi trong Python
  https://www.w3schools.com/python/python_strings.asp
- Các phương thức Xử lý chuỗi trong Python
  https://www.w3schools.com/python/python_strings_methods.asp

Tổng kết lại bài 2

1. Biết cách khai báo BIẾN trong Python
2. Nắm được nguyên tắc đặt tên BIẾN
3. Nắm được phạm vi sử dụng biến (Scope)
4. Nắm được các kiểu dữ liệu của BIẾN, ép kiểu dữ liệu
5. Nắm được cách thực hiện các phép tính trong toán tử số học
6. Nắm được dữ liệu chuỗi và cách xử lý chuỗi