Cho một list các số nguyên n phần tử lst được nhập từ bàn phím. Bạn hãy viết chương trình hiển thị ra màn hình số nhỏ nhất trong list vừa nhập.

**Ví dụ**

* Nếu bạn nhập n = 3, lst = [1, 3, 2] thì chương trình sẽ hiển thị ra 1 (do 1 là giá trị nhỏ nhất trong lst):
* Nếu bạn nhập n = 3, lst = [34, 35, 27] thì chương trình sẽ hiển thị ra 27.

**Lý thuyết**

list trong Python là một kiểu dữ liệu cho phép lưu trữ nhiều kiểu dữ liệu khác. Để khởi tạo một list trong Python bạn có thể sử dụng cặp dấu []. Ví dụ:

# Tạo ra list để lưu trữ các số nguyên

list1 = [1, 2, 3]

# Tạo ra list để lưu trữ các xâu ký tự

list2 = ["Viet", "Tuan", "Duong"]

# Bạn cũng có thể tạo ra một list lưu trữ các kiểu dữ liệu khác nhau

list3 = [7, 3.5, "Codelearn"]

Để truy xuất tới các phần tử trong list bạn dùng toán tử []. Ví dụ:

names = ["Viet", "Dung", "Huong"]

print(names[0])

print(names[1])

print(names[2])

Kết quả khi chạy chương trình:

Viet

Dung

Huong

Lưu ý: names[1] không phải là phần tử đầu tiên của list mà phải là names[0] (do chỉ số của list được bắt đầu từ 0).

Bạn cũng có thể dùng vòng lặp for để duyệt qua các phần tử của list. Ví dụ:

names = ["Viet", "Dung", "Huong"]

for name in names:

print(name)

Để thêm một phần tử vào cuối của list bạn dùng hàm append():

lst = []

lst.append(4)

lst.append(3)

lst.append(6)

print(lst)

Kết quả khi chạy chương trình:

[4, 3, 6]

Đọc tới đây bạn đã biết cách sử dụng list trong python, hãy quay lại phần bài tập và làm thử.

Ngoài kiểu dữ liệu số thì kiểu dữ liệu chuỗi ký tự (str) cũng có các toán tử như +, \*, [], in, not in, ... Trong đó toán tử + dùng để nối chuỗi và toán tử \* dùng để lặp chuỗi. Ví dụ bạn có thể nối 2 chuỗi ký tự giống như sau:

message = "Hello" + " " + "Codelearn"

print(message)

Kết quả khi chạy chương trình:

Hello Codelearn

Bạn có thể lặp lại một chuỗi ký tự giống như sau:

print("Hello " \* 3)

Kết quả khi chạy chương trình:

Hello Hello Hello

Đọc tới đây bạn đã hiểu về cách sử dụng toán tử + để nối chuỗi và toán tử \* để lặp chuỗi. Hãy sử dụng toán tử nối chuỗi để làm bài này.

Bạn chỉ có thể nối 2 chuỗi với nhau chứ không thể nối một chuỗi với 1 số. Ví dụ chương trình sau sẽ báo lỗi:

age = 19

print("Age: " + age)

Để nối được một chuỗi và một số bạn cần phải đưa được số này về dạng chuỗi. Trong ngôn ngữ lập trình Python bạn có thể dùng hàm str() để làm việc này. Ví dụ:

print(type(str(19)))

Kết quả khi chạy chương trình:

<class 'str'>

Ví dụ tiếp về chuyển từ một chuỗi sang một số:

age = 19

print("Age: " + str(age))

Kết quả khi chạy chương trình:

Age: 19

Đọc tới đây bạn đã biết cách chuyển kiểu dữ liệu cho các giá trị, hãy quay lại phần bài tập và làm thử.

Để nhập dữ liệu cho biến từ bàn phím rất đơn giản, bạn sử dụng hàm input() giống như sau:

# Nhập dữ liệu cho biến name từ bàn phím

name = input()

print("Hello " + name)

Sau khi chạy bạn hãy nhập từ bàn phím:

Trung

Và bấm phím **Enter** thì màn hình sẽ hiển thị ra:

Hello Trung

Bạn cần lưu ý rằng khi nhập dữ liệu cho một biến từ bàn phím thì kiểu dữ liệu của biến đó luôn là str (kể cả bạn có nhập giá trị số cho biến đó). Ví dụ chương trình sau:

age = input()

print(type(age))

Nếu bạn nhập 19 và bấm phím **Enter** thì màn hình sẽ hiển thị ra:

<class 'str'>

Đọc tới đây bạn đã biết cách nhập dữ liệu cho biến từ bàn phím, hãy quay lại phần bài tập và làm thử.

Trong Python, dữ liệu nhập từ bàn phím luôn có kiểu là str (hàm input() trả về kiểu str), mà biến kiểu str thì không thể tính toán được. Do đó bạn cần chuyển kiểu dữ liệu của biến age về kiểu int, để làm việc này bạn cần sử dụng hàm int(). Ví dụ chương trình sau:

age = int(input())

age = age + 10

print(age)

print(type(age))

Nếu bạn nhập 23 và bấm **Enter** thì màn hình sẽ hiển thị ra:

33

<class 'int'>

Có thể thấy bạn đã chuyển được chuỗi "23" về số 23.

Đọc tới đây bạn đã biết cách chuyển kiểu dữ liệu từ str về int, hãy quay lại phần bài tập và làm thử.

Để lấy phần dư của một phép chia bạn có thể sử dụng toán tử %. Ví dụ chương trình:

print(5 % 3)

Sẽ hiển thị lên màn hình:

2

Do 2 là phần dư của phép chia 5/3.

Đọc tới đây bạn đã biết cách sử dụng toán tử % để lấy phần dư của phép chia, hãy quay lại phần bài tập và làm thử.

Trong chương trước bạn đã được làm quen với các toán tử như toán tử = +, -, \*, /, %, ... Ở chương này bạn sẽ đi sâu hơn về các toán tử trong Python.

Python xây dựng 7 loại hình tính toán trên toán tử như sau:

* Python Arithmetic Operator (toán tử số học)
* Python Relational Operator (toán tử quan hệ)
* Python Assignment Operator (toán tử gán)
* Python Membership Operator (toán tử membership)
* Python Identity Operator (toán tử identity)
* Python Logical Operator (toán tử logic)
* Python Bitwise Operator (toán tử thao tác bit)

Trong bài này bạn sẽ tìm hiểu về loại toán tử đầu tiên, đó là **Arithmetic Operators**

**Arithmetic Operators**

Đây là toán tử dùng để thao tác với các phép toán như cộng, trừ, nhân, chia, chia dư, ... Xem bảng đầy đủ về các toán tử số học:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Purpose** | **Ví dụ** |
| + | Toán tử cộng 2 giá trị. | 7 + 3 = 10 |
| - | Toán tử trừ 2 giá trị. | 7 - 3 = 4 |
| \* | Toán tử nhân 2 giá trị. | 7 \* 3 = 21 |
| / | Toán tử chia 2 giá trị. | 7 / 3 = 2.(3) |
| // | Toán tử chia lấy phần nguyên của 2 giá trị. | 7 // 3 = 2  10 // 6 = 1 |
| % | Toán tử chia lấy phần dư của 2 giá trị. | 7 % 3 = 1  10 % 6 = 4 |
| \*\* | Toán tử mũ (a\*\*b = ab) | 2 \*\* 3 = 8  5 \*\* 7 = 78125 |

#### **Ví dụ**

x = 15

y = 4

print('x + y =', x+y)

print('x - y =', x-y)

print('x \* y =', x\*y)

print('x / y =', x/y)

print('x % y =', x % y)

print('x // y =', x//y)

print('x \*\* y =', x\*\*y)

Kết quả khi chạy chương trình:

x + y = 19

x - y = 11

x \* y = 60

x / y = 3.75

x % y = 3

x // y = 3

x \*\* y = 50625

Đọc tới đây bạn đã hiểu thêm về các toán tử trong Python

Các toán tử so sánh được dùng để so sánh hai giá trị với nhau. Kết quả của các toán tử này sẽ là True hoăc False (đúng hoặc sai). Các toán tử so sánh sẽ giúp ta kiểm tra xem hai giá trị có bằng, lớn hơn hay nhỏ hơn nhau hay không,... Bảng thể hiển các toán tử so sánh:

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Miêu tả** | **Ví dụ** |
| > | Toán thử lớn hơn - nếu số hạng bên trái lớn hơn số hạng bên phải thì kết quả sẽ là True | 3 > 5 (False) |
| < | Toán tử nhỏ hơn - nếu số hạng bên trái nhỏ hơn số hạng bên phải thì kết quả sẽ là True | 3 < 5 (True) |
| == | Toán tử bằng với - nếu hai số hạng có giá trị bằng nhau thì kết quả sẽ là True. | 3 == 3 (True) |
| != | Toán tử khác bằng - nếu hai số hạng có giá trị khác nhau thì kết quả sẽ là True. | 3 != 3 (False) |
| >= | Toán tử lớn hơn hoặc bằng - nếu số hạng bên trái lớn hơn hoặc bằng số hạng bên phải thì kết quả sẽ là True | 7 >= 6 (True) |
| <= | Toán tử nhỏ hơn hoặc bằng - nếu số hạng bên trái nhỏ hơn hoặc bằng số hạng bên phải thì kết quả sẽ là True | 5 <= 6 (true) |

Ví dụ cụ thể với Python:

x = 10

y = 12

print('x > y is', x > y)

print('x < y is', x < y)

print('x == y is', x == y)

print('x != y is', x != y)

print('x >= y is', x >= y)

print('x <= y is', x <= y)

Kết quả khi chạy chương trình:

x > y is False

x < y is True

x == y is False

x != y is True

x >= y is False

x <= y is True

Đọc tới đây bạn đã hiểu về các toán tử so sánh, hãy quay lại phần bài tập và làm thử.

Để làm được bài này bạn cần hiểu được cách sử dụng các toán tử gán trong Python. Toán tử gán thường được dùng để gán giá trị của vế phải sang cho vế trái. Bảng thể hiện các toán tử gán:

|  |  |  |
| --- | --- | --- |
| **Loại toán tử** | **Mục đích** | **Cách dùng** |
| = | Gán giá trị của vế phải cho vế trái | x = 5 |
| += | Tăng vế trái một phần bằng vế phải sau đó gán giá trị cho vế trái | x += 5  (x = x + 5) |
| -= | Giảm vế trái một phần bằng giá trị vế phải sau đó gán kết quả cho vế trái | x -= 5  (x = x - 5) |
| \*= | Nhân giá trị của vế trái với vế phải sau đó gán kết quả cho vế trái | x \*= 5  (x = x \* 5) |
| /= | Chia giá trị của vế trái cho vế phải sau đó gán kết quả cho vế trái | x /= 5  (x = x / 5) |
| %= | Chia giá trị của vế trái cho vế phải sau đó gán phần dư cho vế trái (chia lấy dư) | x %= 5  (x = x % 5) |
| //= | Phép chia lấy phần nguyên. | x //= 5  (x = x // 5) |
| \*\*= | Lấy vế trái lũy thừa với bậc là giá trị vế phải sau đó gán kết quả cho vế trái | x \*\*= 5  (x = x \*\* 5) |
| &= | Thực hiện phép toán của toán tử AND cho 2 vế sau đó gán cho vế trái | x &= 5  (x = x & 5) |
| |= | Thực hiện phép toán của toán tử OR cho 2 vế sau đó gán cho vế trái | x |= 5  (x = x | 5) |
| ^= | Thực hiện phép toán của toán tử XOR 2 vế sau đó gán cho vế trái | x ^= 5  (x = x ^ 5) |
| >>= | Thực hiện phép toán dịch phải của vế phải sau đó gán cho vế trái | x >>= 5  (x = x >> 5) |
| <<= | Thực hiện phép toán dịch trái của vế phải sau đó gán cho vế trái | x <<= 5  (x = x << 5) |

Để làm được bài này bạn cần biết về toán tử membership. Toán tử membership là toán tử dùng để kiểm xem một giá trị có nằm trong giá trị khác hay không.

in và not in là hai toán tử membership với cách sử dụng rất đơn giản. Ví dụ bạn có thể kiểm tra xem một xâu có nằm trong một xâu khác không giống như sau:.

print("Code" in "Codelearn")

print("Py" not in "Python")

Kết quả khi chạy chương trình:

True

False

biến có đang trỏ tới cùng một đối tượng hay không. Với các kiểu dữ liệu như int, str, float,... thì toán tử này tương đương với toán tử ==. Bạn sẽ được học về sự khác nhau giữa hai toán tử này ở các bài sau.

Trong Python, is và is not chính là 2 toán tử định danh. Ví dụ:

a = 5

b = 7

print(a is b)

print(a is not b)

Kết quả khi chạy chương trình:

False

True

Toán tử logical (logic) not, or và and là các toán tử được dùng để kết hợp các mệnh đề lại với nhau. Bảng thể hiện toán tử logic:

|  |  |  |
| --- | --- | --- |
| **Loại toán tử** | **Mục đích** | **Cách dùng** |
| and | Trả về True nếu hai điều kiện cùng đúng, ngược lại trả về False | a and b |
| or | Trả về True nếu có ít nhất một điều kiện đúng, ngược lại nếu cả hai điều kiện đều sai thì trả về False | a or b |
| not | Toán tử phủ định, toán tử này trả về False nếu điều kiện là True, ngược lại nếu điều kiện là False thì trả về True | not a |

Để làm được bài này bạn cần biết cách sử dụng đề if-else. Mệnh đề if-else được dùng để kiểm tra một điều kiện có đúng hay không, nếu đúng thì khối lệnh bên trong if sẽ được thực thi, ngược lại nếu điều kiện là sai thì khối lệnh bên trong else sẽ được thực thi:

if condition:

# Nếu condition đúng thì khối lệnh này sẽ được thực thi

else:

# Nếu condition sai thì khối lệnh này sẽ được thực thi

Lưu ý: khối lệnh trong if và else chính là các đoạn code được thụt lề vào trong (ở các ngôn ngữ lập trình khác thì khối lệnh được xác định bằng cặp dấu {} nhưng đối với Python thì khối lệnh được xác định bằng việc thụt lề).

Ví dụ về chương trình sử dụng mệnh đề if-else để kiểm tra tính chẵn lẻ của một số được nhập từ bàn phím:

n = int(input())

if n % 2 == 0:

# Nếu n là số chẵn thì hiển thị ra màn hình n is an even number

print("n is an even number")

else:

# Ngược lại, nếu n không là số chẵn thì hiển thị ra màn hình n is an odd number

print("n is an odd number")

Chương trình trên cũng có thể tách thành 2 mệnh đề if giống như sau:

n = int(input())

if n % 2 == 0:

# Nếu n là số chẵn thì hiển thị ra màn hình n is an even number

print("n is an even number")

if n % 2 != 0:

# Nếu n là số lẻ thì hiển thị ra màn hình n is an odd number

print("n is an odd number")

#### **Toán tử 3 ngôi trong Python**

Toán tử ba ngôi cũng được dùng để kiểm tra các điều kiện giống như mệnh đề if-else nhưng nó đơn giản hơn ở chỗ có thể viết gọn thành một dòng. Toán tử 3 ngồi có thể thay thế cho mệnh đề if-else khi cần kiểm tra các điều kiện đơn giản.

Ví dụ về chương trình so sánh giá trị 2 số sử dụng toán tử 3 ngồi:

# Python program to demonstrate nested ternary operator

a, b = 10, 20

print ("Both a and b are equal" if a == b else "a is greater than b"

if a > b else "b is greater than a")

Nếu sử dụng mệnh đề if-else để so sánh giá trị 2 số thì sẽ như sau:

a, b = 10, 20

if a != b:

if a > b:

print("a is greater than b")

else:

print("b is greater than a")

else:

print("Both a and b are equal")

Đọc tới đây bạn đã biết cách sử dụng mệnh đề if-else và toán tử 3 ngôi, hãy thử làm bài tập này bằng 2 cách.

Như ở chương trước bạn đã được học, toán tử and có thể được dùng để kết hợp các điều kiện lại. Ví dụ bạn có thể kiểm tra xem một số có nằm trong đoạn [10, 20] không giống như sau:

n = int(input())

if n >= 10 and n <= 20:

print("n is in the range [10, 20]")

else:

print("n is not in the range [10, 20]")

Đọc tới đây bạn đã biết cách sử dụng toán tử and để kết hợp với mệnh đề if, hãy quay lại phần bài tập và làm thử

Để làm được bài này bạn cần biết cách sử dụng vòng lặp trong Python. Vòng lặp được hiểu đơn giản là lặp lại một đoạn code nhiều lần. Trong Python có hỗ trợ 2 loại vòng lặp là vòng lặp while và vòng lặp for.

**Vòng lặp while**

Vòng lặp while được dùng để lặp lại một hành động cho tới khi điều kiện lặp không còn thỏa mãn nữa. Cú pháp của vòng lặp while:

while condition:

# Khối lệnh này sẽ được thực thi nếu condition còn đúng

Ví dụ về chương trình hiển thị ra màn hình các số từ 1 tới 5 sử dụng vòng lặp while:

i = 1

while i <= 5:

print(i)

i += 1

Kết quả khi chạy chương trình:

1

2

3

4

5

Có thể thấy cách sử dụng vòng lặp while rất đơn giản, bạn có thể sử dụng vòng lặp while để tính tổng các số từ 1 tới n giống như sau:

n = int(input())

i = 1

answer = 0

while i <= n:

answer += i

i += 1

print(answer)

**Vòng lặp for**

Không giống với vòng lặp while, vòng lặp for được dùng để lặp qua một tập hợp cho trước, vòng lặp for thường được sử dụng với hàm range(). Ví dụ:

for i in range(1, 5):

print(i)

Kết quả khi chạy chương trình:

1

2

3

4

Giải thích: range() chính là hàm trả về một tập hợp, như ở ví dụ trên thì range(1, 5) sẽ trả về một tập hợp chứa các số từ 1 tới 4.

Ngoài ra, bạn có thể sử dụng vòng lặp for để duyệt qua các ký tự của một xâu. Ví dụ:

name = "Codelearn"

for c in name:

print(c)

Kết quả khi chạy chương trình:

C

o

d

e

l

e

a

r

n

Bài này sẽ giúp bạn hiểu về các câu lệnh điều khiển vòng lặp.

**Câu lệnh Break**

Câu lệnh break được dùng để thoát khỏi vòng lặp, khi chương trình gặp câu lệnh break thì vòng lặp sẽ không được thực thi tiếp. Ví dụ:

for i in range(1, 11):

if i == 6:

break

print(i)

Kết quả khi chạy chương trình:

1

2

3

4

5

Có thể thấy chương trình trên hiển thị ra màn hình các số từ 1 tới 5 thay vì từ 1 tới 10 do khi i == 6 thì chương trình gặp câu lệnh break và vòng lặp sẽ không được thực thi tiếp.

**Câu lệnh continue**

Khi gặp câu lệnh continue trong vòng lặp, các đoạn code bên dưới câu lênh này trong vòng lặp sẽ không được thực thi. Ví dụ:

for i in range(1, 20):

if i % 2 == 0:

continue

print(i)

Kết quả khi chạy chương trình:

1

3

5

7

9

11

13

15

17

19

Giải thích: chương trình trên không hiển thị ra màn hình các số từ 1 tới 20 do khi i % 2 == 0 (i là số chẵn) thì chương trình sẽ gặp câu lệnh continue và do câu lệnh print(i) nằm dưới câu lệnh này nên sẽ không được thực thi.

Đọc tới đây bạn đã hiêu về các câu lệnh điều khiển vòng lặp, hãy quay lại phần bài tập và làm thử.

Bạn có thể sử dụng hàm round() để làm tròn các số thực. Cú pháp hàm round:

round(number, ndigits)

Trong đó number là số cần làm tròn, ndigits là số chữ số sau dấu phẩy cần làm tròn. Ví dụ:

print(round(1.23, 1))

print(round(2.665, 2))

print(round(2.673567, 4))

Kết quả khi chạy chương trình:

1.2

2.67

2.6736

Đọc tới đây bạn đã biết cách sử dụng hàm round() để làm tròn số, hãy quay lại phần bài tập và làm thử.

list trong Python là một kiểu dữ liệu cho phép lưu trữ nhiều kiểu dữ liệu khác. Để khởi tạo một list trong Python bạn có thể sử dụng cặp dấu []. Ví dụ:

# Tạo ra list để lưu trữ các số nguyên

list1 = [1, 2, 3]

# Tạo ra list để lưu trữ các xâu ký tự

list2 = ["Viet", "Tuan", "Duong"]

# Bạn cũng có thể tạo ra một list lưu trữ các kiểu dữ liệu khác nhau

list3 = [7, 3.5, "Codelearn"]

Để truy xuất tới các phần tử trong list bạn dùng toán tử []. Ví dụ:

names = ["Viet", "Dung", "Huong"]

print(names[0])

print(names[1])

print(names[2])

Kết quả khi chạy chương trình:

Viet

Dung

Huong

Lưu ý: names[1] không phải là phần tử đầu tiên của list mà phải là names[0] (do chỉ số của list được bắt đầu từ 0).

Bạn cũng có thể dùng vòng lặp for để duyệt qua các phần tử của list. Ví dụ:

names = ["Viet", "Dung", "Huong"]

for name in names:

print(name)

Để thêm một phần tử vào cuối của list bạn dùng hàm append():

lst = []

lst.append(4)

lst.append(3)

lst.append(6)

print(lst)

Kết quả khi chạy chương trình:

[4, 3, 6]

Đọc tới đây bạn đã biết cách sử dụng list trong python, hãy quay lại phần bài tập và làm thử.

Bài này sẽ giúp bạn biết về các hàm thường được sử dụng trong list.

**1. Hàm len**

Đây là hàm trả về số phần tử có trong list. Ví dụ:

lst = [2, 3, 1]

print(len(lst))

Kết quả khi chạy chương trình:

3

Bạn có thể tận dụng hàm len() để hỗ trợ duyệt qua các phần tử trong list:

lst = [2, 3, 1]

for i in range(len(lst)):

print(lst[i])

**2. Hàm max, min**

Đây là hai hàm được dùng để trả về phần tử lớn nhất và nhỏ nhất trong list. Ví dụ:

lst = [2, 3, 1]

print(max(lst))

print(min(lst))

Kết quả khi chạy chương trình:

3

1

**3. Hàm insert**

Đây là hàm dùng để thêm một phần tử vào một ví trí trong list:

vowels = ['a', 'e', 'i', 'u']

# Chèn xâu 'o' vào vị trí thứ 4 trong list vowels

vowels.insert(3, 'o')

print(vowels)

Kết quả khi chạy chương trình:

['a', 'e', 'i', 'o', 'u']

**4. Hàm remove**

Hàm này dùng để xóa một phần tử khỏi list:

lst = ['A', 'B', 'C']

lst.remove('A')

print(lst)

Kết quả khi chạy chương trình:

['B', 'C']

**5. Hàm pop**

Hàm pop() được dùng để xóa một phần tử với chỉ số cho trước trong list:

lst = ['A', 'B', 'C']

# Xóa phần tử thứ 2 khỏi list

lst.pop(1)

print(lst)

Kết quả khi chạy chương trình:

['A', 'C']

**6. Hàm sort**

Hàm này được dùng để sắp xếp các phần tử trong list theo một thứ tự nhất định. Ví dụ:

lst = [4, 5, 3, 7, 6, 1]

# Sắp xếp các phần tử trong list theo thứ tự tăng dần

lst.sort()

print(lst)

# Sắp xếp các phần tử trong list theo thứ tự giảm dần

lst.sort(reverse=True)

print(lst)

Kết quả khi chạy chương trình:

[1, 3, 4, 5, 6, 7]

[7, 6, 5, 4, 3, 1]

**7. Hàm reverse**

Đây là hàm dùng để đảo ngược list:

lst = [4, 5, 3, 7, 6, 1]

lst.reverse()

print(lst)

Kết quả khi chạy chương trình:

[1, 6, 7, 3, 5, 4]

**8. Hàm count**

Đây là hàm dùng để đếm số lần xuất hiện của một thành phần trong list:

lst = [6, 2, 3, 8, 2]

print(lst.count(2))

Kết quả khi chạy chương trình:

2

**9. Hàm clear**

Đây là hàm dùng để xóa hết các phần tử bên trong list:

lst = [1, 2, 3]

lst.clear()

print(lst)

Kết quả khi chạy chương trình:

[]

Đọc tới đây bạn đã biết về các hàm hay được sử dụng trong list, hãy quay lại phần bài tập và làm thử.

Ở trong các chương trước bạn đã được làm quen với chuỗi, trong chương này bạn sẽ được học một vài kỹ thuật xử lý chuỗi.

Trong Python, chuỗi là một tập hợp các kí tự Unicode. Tuy nhiên, Python không có kiểu kí tự (character) giống các ngôn ngữ lập trình khác, thay vào đó bạn có thể hiểu rằng ký tự là một xâu có độ dài bằng 1.

Để khai báo một chuỗi trong Python bạn có thể sử dụng một cặp dấu nháy đơn hoặc nháy đôi. Ví dụ "Codelearn" và 'Codelearn' đều là một chuỗi. Ngoài ra bạn cũng có thể sử dụng 3 cặp nháy đơn, nháy đôi để tạo ra một chuỗi trên nhiều dòng. Ví dụ:

s = """

Banana

Apple

Orange

"""

print(s)

Kết quả khi chạy chương trình:

Banana

Apple

Orange

Trong Python, các chuỗi được viết bằng 3 dấu nháy thường được sử dụng như các comments nếu như nó không được gán cho một biến cụ thể nào.

#### Một số hàm và phương thức để xử lý chuỗi trong Python

**1. Hàm len()**

Đây là hàm trả về độ dài của chuỗi:

s = "abc"

print(len(s))

s = "abcd"

print(len(s))

Kết quả khi chạy chương trình:

3

4

**2. Phương thức lower()**

Đây là phương thức được dùng để chuyển 1 chuỗi về dạng in thường:

s = "CODELEARN123"

print(s.lower())

Kết quả khi chạy chương trình:

codelearn123

**3. Phương thức upper()**

Đây là phương thức được dùng để chuyển 1 chuỗi về dạng in hoa:

s = "codelearn123"

print(s.upper())

Kết quả khi chạy chương trình:

CODELEARN123

**4. Phương thức isalnum()**

Đây là phương thức được dùng để kiểm tra xem một xâu có chỉ chứa các ký tự chữ và số hay không. Ví dụ:

s = "codelearn2020"

print(s.isalnum())

s = "codelearn2020.io"

# Kết quả sẽ là False do chuỗi s chứa ký tự .

print(s.isalnum())

Kết quả khi chạy chương trình:

True

False

**5. Phương thức isalpha()**

Phương thức này được dùng để kiểm tra xem một chuỗi có chứa toàn các ký tự chữ không:

s = "codelearn"

print(s.isalpha())

# Kết quả sẽ là False do chuỗi s chứa số 2020

s = "codelearn2020"

print(s.isalpha())

Kết quả khi chạy chương trình:

True

False

**6. Phương thức isnumeric()**

Phương thức này dùng để kiểm tra xem một xâu có chứa toàn các ký tự số hay không:

s = "2020"

print(s.isnumeric())

s = "c2020"

print(s.isnumeric())

Kết quả khi chạy chương trình:

True

False

**7. Phương thức split()**

Phương thức này được dùng để cắt một chuỗi ra thành list các chuỗi khác dựa trên một phần tử trong chuỗi đầu vào:

s = "Welcome to Codelearn.io!"

print(s.split(" "))

s = "A1B1C1D1E1"

print(s.split("1"))

Kết quả khi chạy chương trình:

['Welcome', 'to', 'Codelearn.io!']

['A', 'B', 'C', 'D', 'E', '']

**8. Phương thức join()**

Phương thức này được dùng để nối một tập hợp thành một chuỗi sử dụng kí tự cho trước. Ví dụ:

lst = ["Welcome", "to", "Codelearn.io!"]

print(" ".join(lst))

lst = ["A", "B", "C"]

print("-".join(lst))

Kết quả khi chạy chương trình:

Welcome to Codelearn.io!

A-B-C

Bạn có thể sử dụng hàm split() và hàm join() để loại bỏ các khoảng trắng thừa trong chuỗi. Ví dụ:

message = " Welcome to Codelearn.io! "

print(" ".join(message.split()))

Kết quả khi chạy chương trình:

Welcome to Codelearn.io!

**9. Phương thức replace()**

Phương thức này được dùng để thay thế các chuỗi con tìm thấy thành chuỗi con mới. Ví dụ:

name = "Cod3l3arn"

print(name.replace("3", "e"))

Kết quả khi chạy chương trình:

Codelearn

Đọc tới đây bạn đã biết cách sử dụng các hàm và phương thức có sẵn để xử lý chuỗi, hãy quay lại phần bài tập và làm thử.

Bạn có thể lấy ra một ký tự trong chuỗi thông qua chỉ số giống như lấy ra một phần tử trong list. Ví dụ:

name = "Codelearn"

print(name[0])

Kết quả khi chạy chương trình:

C

Ngoài ra, trong ngôn ngữ lập trình Python bạn còn có thể truy xuất tới chỉ số âm của chuỗi. Vị trí -1 sẽ thể hiện cho ký tự cuối cùng trong chuỗi, tương tự với -2, -3, -4, ... Ví dụ:

s = 'Python String'

# s[0] là phần tử đầu tiên trong chuỗi

print(s[0])

# s[-1] là phần tử đầu cuối cùng trong chuỗi

print(s[-1])

# s[-2] là phần tử đứng trước phần tử cuối cùng trong chuỗi

print(s[-2])

Kết quả khi chạy chương trình:

P

g

n

**Cắt chuỗi trong Python (Slice a string in Python)**

Để lấy ra một dãy các kí tự liên tiếp trong chuỗi, bạn có thể sử dụng phương pháp slicing giống như sau:

s = 'Python String'

print(s[0:2])

print(s[3:5])

print(s[7:])

print(s[:6])

print(s[7:-4])

Kết quả khi chạy chương trình:

Py

ho

String

Python

St

Đọc tới đây bạn đã biết cách cắt chuỗi, hãy quay lại phần bài tập và làm thử.

Trước khi làm bài này bạn hãy xem thêm một vài toán tử để thao tác với chuỗi:

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Mô tả** | **Ví dụ** |
| [] | Slice - Trả về kí tự trong chuỗi | x="Python"  print(x[1]) |
| [:] | Range slice-Trả về dãy kí tự trong chuỗi | x="Python"  print(x[1:3]) |
| in | Kiểm tra xem một chuỗi có nằm trong chuỗi khác không | x="Python"  print("u" in x) |
| not in | Kiểm tra xem một chuỗi có không nằm trong chuỗi khác không | x="Python"  print("l" not in x) |
| % | Hỗ trợ định dạng chuỗi | x = 12.3456789  print("The value of x is %3.2f" %x) |
| + | Cộng hai chuỗi bất kì | x="Code"  y="Learn"  print(x+y) |
| \* | Lặp lại chuỗi bất kì cho trước | x="CodeLearn"  print(x\*2) |

Trong các chương trước bạn đã biết cách sử dụng các hàm có sẵn như hàm len(), upper(), lower(), sort(), ... qua bài này bạn sẽ hiểu được cách tạo và sử dụng hàm.

Hàm được hiểu đơn giản là 1 đoạn code, khi bạn gọi hàm thì thực ra là bạn đang gọi tới đoạn code này (bạn cũng có thể hiểu hàm là viết tắt của 1 đoạn code). Để khai báo một hàm bạn dùng từ khóa def giống như sau:

def <tên\_hàm>:

# Các đoạn code trong thân hàm

Ví dụ về hàm hiển thị ra màn hình các số từ 1 tới 10:

def show():

for i in range(1, 11):

# end = " " có nghĩa là sử dụng hàm print và không xuống dòng

print(i, end=" ")

print()

# gọi tới hàm show

show()

show()

Kết quả khi chạy chương trình:

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

Có thể thấy cách tạo và sử dụng hàm rất đơn giản, chương trình trên hiển thị ra 2 lần các số từ 1 tới 10 do gọi tới hàm show() 2 lần.

Ngoài ra bạn cũng có thể truyền các tham số vào hàm và trả về kết quả bằng từ khóa return <giá\_trị\_trả\_về>. Ví dụ về hàm trả về tổng của một list:

def sum\_of\_list(lst):

answer = 0

for v in lst:

answer += v

return answer

print(sum\_of\_list([3, 4, 2]))

print(sum\_of\_list([8, 4, 7]))

print(sum\_of\_list([1, 2, 3]))

Kết quả khi chạy chương trình:

9

19

6

Ví dụ khác về hàm trả về số các số chẵn trong list:

def count\_even(lst):

count = 0

for v in lst:

if v % 2 == 0:

count += 1

return count

print(count\_even([3, 4, 2]))

print(count\_even([2, 4, 6]))

print(count\_even([7, 3, 2]))

Kết quả khi chạy chương trình:

2

3

1

Đọc tới đây bạn đã biết cách tạo và sử dụng hàm, hãy quay lại phần bài tập và làm thử.