Lập trình với Python

Chương 2: Các khái niệm

Tài liệu tham khảo

1. Hill, C. (2020). Learning scientific programming with Python. Cambridge University Press.
2. Stephenson, B. (2019). The Python Workbook 2nd. Springer.
3. Pine, D. J. (2019). Introduction to Python for science and engineering. CRC Press.
4. Kong, Q., Siauw, T., & Bayen, A. M. (2021) Python Programming and Numerical Methods - A Guide for Engineers and Scientists. Elsevier Inc.

Định danh

- Định danh là tên được đặt cho một biến, hàm, lớp hoặc mô-đun. Định danh có thể là một hoặc nhiều ký tự.
- Định danh có thể là sự kết hợp của các chữ cái viết thường (a đến z) hoặc viết hoa (A đến Z) hoặc các chữ số (0 đến 9) hoặc một dấu gạch dưới (_).
- Định danh Python có thể bắt đầu bằng bảng chữ cái (A-Z và a-z và _). Định danh không thể bắt đầu bằng một chữ số như được phép ở ngôn ngữ khác.

Từ khóa

- Từ khóa là danh sách các từ dành riêng có nghĩa được xác định trước.
- Từ khóa là từ đặc biệt mà người lập trình không thể sử dụng làm định danh cho các biến, hàm, hằng số hoặc với bất kỳ tên định danh nào.
- Cố gắng sử dụng một từ khóa làm tên định danh sẽ gây ra lỗi.
- Bảng sau đây hiển thị các từ khóa Python.

List of Keywords in Python

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

Câu lệnh và biểu thức

- Một câu lệnh là một lệnh mà trình thông dịch Python có thể thực thi. Chương trình Python bao gồm một chuỗi các câu lệnh.
- Biểu thức là một sự sắp xếp các giá trị và toán tử để tạo ra một giá trị mới.
- Biểu thức cũng là câu lệnh. Một giá trị duy nhất hoặc một biến duy nhất hoặc sự kết hợp của biến, toán tử và giá trị đều là biểu thức.

Biến

- Biến là một trình giữ chỗ được đặt tên để giữ bất kỳ loại dữ liệu nào mà chương trình có thể sử dụng để gán và sửa đổi trong quá trình thực thi.
- Trong Python, không cần khai báo biến một cách rõ ràng bằng cách chỉ định biến đó là số nguyên hay số thực hay bất kỳ kiểu nào khác.
- Để xác định một biến mới trong Python, chúng ta chỉ cần gán một giá trị cho một tên.

Biến

- Thực hiện theo các quy tắc được đề cập bên dưới để tạo tên biến hợp pháp trong Python.

✓ Tên biến có thể bao gồm bất kỳ số lượng chữ cái, dấu gạch dưới và chữ số nào.

✓ Biến không nên bắt đầu bằng một số.

✓ Từ khóa Python không được phép làm tên biến.

✓ Tên biến phân biệt chữ hoa chữ thường. Ví dụ: máy tính và Máy tính là các biến khác nhau.

Biến

- Định dạng chung để gán giá trị cho các biến như sau:

variable_name = expression

- Dấu bằng (=) còn được gọi là toán tử gán đơn giản được sử dụng để gán giá trị cho biến.
- Giá trị đó được lưu trữ trong biến trên việc thực hiện câu lệnh gán.

Biến

```
1. >>> number =100
2. >>> miles =1000.0
3. >>> name ="Python"
4. >>> number
100
5. >>> miles
1000.0
6. >>> name
‘Python’
```

Biến

```
1. >>> a = b = c =1
2. >>> a
1
3. >>> b
1
4. >>> c
1
```

Toán tử

- Toán tử là các ký hiệu, chẳng hạn như +, -, =,> và <, thực hiện một số phép toán hoặc hoạt động logic để thao tác các giá trị dữ liệu và tạo ra một kết quả dựa trên một số quy tắc.
- Một toán tử thao tác các giá trị dữ liệu được gọi là toán hạng.
- Hãy xem xét biểu thức,

  >>> 4 + 6 (trong đó 4 và 6 là toán hạng và + là toán tử.)

Toán tử số học

- Toán tử số học được sử dụng để thực hiện các phép tính số học như cộng, trừ, chia, nhân, v.v. Bảng sau đây cho thấy tất cả các các toán tử số học.
- Ví dụ

```
1. >>> 10+35
   45
2. >>> –10+35
   25
3. >>> 4*2
   8
4. >>> 4**2
   16
5. >>> 45/10
   4.5
6. >>> 45//10.0
   4.0
7. >>> 2025%10
   5
8. >>> 2025//10
   202
```

Toán tử số hoc

List of Arithmetic Operators

Operator	Operator Name	Description	Example
+	Addition operator	Adds two operands, producing their sum.	p + q = 5
–	Subtraction operator	Subtracts the two operands, producing their difference.	p – q = –1
*	Multiplication operator	Produces the product of the operands.	p * q = 6
/	Division operator	Produces the quotient of its operands where the left operand is the dividend and the right operand is the divisor.	q / p = 1.5
%	Modulus operator	Divides left hand operand by right hand operand and returns a remainder.	q % p = 1
**	Exponent operator	Performs exponential (power) calculation on operators.	p**q = 8
//	Floor division operator	Returns the integral part of the quotient.	9//2 = 4 and 9.0//2.0 = 4.0

Note: The value of p is 2 and q is 3.

Toán tử gán

- Các toán tử gán được sử dụng để gán các giá trị được tạo sau khi toán hạng bên phải với toán hạng bên trái. Thao tác gán luôn hoạt động từ phải sang trái. Các toán tử gán là toán tử gán đơn giản hoặc phép gán ghép các toán tử.
- Phép gán đơn giản được thực hiện với dấu bằng (=) và chỉ cần gán giá trị của toán hạng bên phải của nó với biến ở bên trái.

List of Assignment Operators

Operator	Operator Name	Description	Example
=	Assignment	Assigns values from right side operands to left side operand.	z = p + q assigns value of p + q to z
+=	Addition Assignment	Adds the value of right operand to the left operand and assigns the result to left operand.	z += p is equivalent to z = z + p
−=	Subtraction Assignment	Subtracts the value of right operand from the left operand and assigns the result to left operand.	z −= p is equivalent to z = z − p
*=	Multiplication Assignment	Multiplies the value of right operand with the left operand and assigns the result to left operand.	z *= p is equivalent to z = z * p
/=	Division Assignment	Divides the value of right operand with the left operand and assigns the result to left operand.	z /= p is equivalent to z = z / p
**=	Exponentiation Assignment	Evaluates to the result of raising the first operand to the power of the second operand.	z**= p is equivalent to z = z ** p
//=	Floor Division Assignment	Produces the integral part of the quotient of its operands where the left operand is the dividend and the right operand is the divisor.	z //= p is equivalent to z = z // p
%=	Remainder Assignment	Computes the remainder after division and assigns the value to the left operand.	z %= p is equivalent to z = z % p

Toán tử gán

```
1. >>> x = 5
2. >>> x = x + 1
3. >>> x
   6
```

```
1. >>> p = 10
2. >>> q = 12
3. >>> q += p
4. >>> q
   22
5. >>> q *= p
6. >>> q
   220
7. >>> q /= p
8. >>> q
   22.0
```

```
9. >>> q %= p
10. >>> q
    2.0
11. >>> q **= p
12. >>> q
    1024.0
13. >>> q //= p
14. >>> q
    102.0
```

Toán tử so sánh

- Khi giá trị của hai toán hạng được so sánh thì các toán tử so sánh được sử dụng.
- Đầu ra của các toán tử so sánh này luôn là giá trị Boolean, True hoặc False.
- Toán hạng có thể là Numbers hoặc Strings hoặc Boolean giá trị. Chuỗi là chữ cái so sánh bằng chữ cái sử dụng các giá trị ASCII của chúng, do đó, “P” nhỏ hơn “Q” và “Aston” lớn hơn"Asher". Bảng sau hiển thị tất cả các toán tử so sánh.

Toán tử so sánh

List of Comparison Operators

Operator	Operator Name	Description	Example
==	Equal to	If the values of two operands are equal, then the condition becomes True.	(p == q) is not True.
!=	Not Equal to	If values of two operands are not equal, then the condition becomes True.	(p != q) is True
>	Greater than	If the value of left operand is greater than the value of right operand, then condition becomes True.	(p > q) is not True.
<	Lesser than	If the value of left operand is less than the value of right operand, then condition becomes True.	(p < q) is True.
>=	Greater than or equal to	If the value of left operand is greater than or equal to the value of right operand, then condition becomes True.	(p >= q) is not True.
<=	Lesser than or equal to	If the value of left operand is less than or equal to the value of right operand, then condition becomes True.	(p <= q) is True.

Note: The value of p is 10 and q is 20.

Toán tử so sánh

1. >>>10 == 12
   False
2. >>>10 != 12
   True
3. >>>10 < 12
   True
4. >>>10 > 12
   False
5. >>>10 <= 12
   True
6. >>>10 >= 12
   False
7. >>> "P" < "Q"
   True
8. >>> "Aston" > "Asher"
   True
9. >>> True == True
   True

Toán tử Logic

- Các toán tử logic được sử dụng để so sánh hoặc phủ định các giá trị logic của các toán hạng của chúng và để trả về giá trị logic kết quả.
- Giá trị của các toán hạng mà toán tử logic hoạt động đánh giá True hoặc False. Kết quả của toán tử logic luôn là giá trị Boolean, True hoặc False.
- Bảng sau hiển thị tất cả các toán tử logic.

Toán tử Logic

List of Logical Operators

Operator	Operator Name	Description	Example
and	Logical AND	Performs AND operation and the result is True when both operands are True	p and q results in False
or	Logical OR	Performs OR operation and the result is True when any one of both operand is True	p or q results in True
not	Logical NOT	Reverses the operand state	not p results in False

Note: The Boolean value of p is True and q is False.

Toán tử Logic

Boolean Logic Truth Table

P	Q	P and Q	P or Q	Not P
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

```
1. >>> True and False
   False
2. >>> True or False
   True
3. >>> not(True) and False
   False
4. >>> not(True and False)
   True
5. >>> (10 < 0) and (10 > 2)
   False
6. >>> (10 < 0) or (10 > 2)
   True
7. >>> not(10 < 0) or (10 > 2)
   True
8. >>> not(10 < 0 or 10 > 2)
   False
```

Toán tử Bitwise

- Toán tử bitwise coi các toán hạng của chúng như một chuỗi các bit (số 0 và số 1) và thực hiện hoạt động từng bit.
- Ví dụ: số thập phân có biểu diễn nhị phân của 1010. Các toán tử bitwise thực hiện các hoạt động của họ trên các biểu diễn nhị phân như vậy, nhưng chúng trả về các giá trị số Python tiêu chuẩn.
- Bảng sau hiển thị tất cả các toán tử bitwise.

List of Bitwise Operators

Operator	Operator Name	Description	Example
&	Binary AND	Result is one in each bit position for which the corresponding bits of both operands are 1s.	p & q = 12 (means 0000 1100)
|	Binary OR	Result is one in each bit position for which the corresponding bits of either or both operands are 1s.	p | q = 61 (means 0011 1101)
^	Binary XOR	Result is one in each bit position for which the corresponding bits of either but not both operands are 1s.	(p ^ q) = 49 (means 0011 0001)
~	Binary Ones Complement	Inverts the bits of its operand.	(~p) = −61 (means 1100 0011 in 2's complement form due to a signed binary number.
<<	Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	p << 2 = 240 (means 1111 0000)
>>	Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	p >> 2 = 15 (means 0000 1111)

Note: The value of p is 60 and q is 13.

Toán tử Bitwise

Bitwise Truth Table

P	Q	P & Q	P | Q	P ^ Q	~ P
0	0	0	0	0	1
0	1	0	1	1	
1	0	0	1	1	0
1	1	1	1	0	

Bitwise and (&)

a	=	0011 1100	→	(60)
b	=	0000 1101	→	(13)
a & b	=	0000 1100	→	(12)

Bitwise or (|)

a	=	0011 1100	→	(60)
b	=	0000 1101	→	(13)
a | b	=	0011 1101	→	(61)

Bitwise exclusive or (^)

a	=	0011 1100	→	(60)
b	=	0000 1101	→	(13)
a ^ b	=	0011 0001	→	(49)

One's Complement (~)

a	=	0011 1100	→	(60)
~ a	=	1100 0011	→	(-61)

Binary left shift (<<)

a	=	0011 1100	→	(60)
a << 2	=	1111 0000	→	(240)

left shift of 2 bits

Binary right shift (>>)

a	=	0011 1100	→	(60)
a >> 2	=	0000 1111	→	(15)

right shift of 2 bits

Tính ưu tiên và liên kết

- Mức độ ưu tiên của toán tử xác định cách thức mà các toán tử được phân tích cú pháp liên quan đến lẫn nhau. Các toán tử có mức độ ưu tiên cao hơn trở thành toán hạng của các toán tử có mức độ ưu tiên thấp hơn.
- Tính liên kết xác định cách thức mà các toán tử có cùng mức độ ưu tiên được phân tích cú pháp. Hầu như tất cả các toán tử đều có sự kết hợp từ trái sang phải. Mức độ ưu tiên của toán tử được liệt kê trong bảng sau mức độ ưu tiên cao nhất đến mức độ ưu tiên thấp nhất.

Operator Precedence in Python

Operators	Meaning
()	Parentheses
**	Exponent
+x, −x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, −	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisons, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

Tính ưu tiên và liên kết

```
1. >>> 2 + 3 * 6
   20
2. >>> (2 + 3) * 6
   30
3. >>> 6 * 4 / 2
   12.0
```

Kiểu dữ liệu

- Kiểu dữ liệu như số và ký tự sẽ được lưu trữ và thao tác trong chương trình. Các kiểu dữ liệu cơ bản của Python là Number, Boolean, String, None.
- Số nguyên, số dấu phẩy động và số phức thuộc danh mục số trong Python.
- Chúng được định nghĩa là lớp int, float và complex trong Python.

Kiểu Number

- Số nguyên có thể là bất kỳ chiều dài.
- Số dấu phẩy động chính xác lên đến 15 chữ số thập phân. Số nguyên và dấu phẩy động được phân tách bằng dấu thập phân.
- Số phức được viết dưới dạng, x + yj, trong đó x là phần thực và y là phần ảo.

Kiểu Boolean

- Booleans chúng rất cần thiết khi bạn bắt đầu sử dụng câu điều kiện.
- Một điều kiện thực sự chỉ là một câu hỏi Yes hoặc No, câu trả lời cho câu hỏi đó là giá trị Boolean, True hoặc False.
- Các giá trị Boolean, True và False được coi là từ dành riêng.

Kiểu String

- Chuỗi bao gồm một chuỗi gồm một hoặc nhiều ký tự, có thể bao gồm các chữ cái, số và các loại ký tự khác. Một chuỗi cũng có thể chứa khoảng trắng.
- Có thể sử dụng dấu ngoặc đơn hoặc dấu ngoặc kép để biểu diễn chuỗi và nó còn được gọi là một chuỗi ký tự.
- Chuỗi đa dòng có thể được biểu thị bằng cách sử dụng dấu ngoặc kép, '''hoặc """. Đây là các giá trị cố định, không phải là biến mà bạn thực sự cung cấp trong tập lệnh của mình.

Kiểu String

```
1. >>> s = 'This is single quote string'
2. >>> s = "This is double quote string"

3. >>> s = '''This is
                    Multiline
                              string'''
4. >>> s = "a"
```

Kiểu None

- None là một kiểu dữ liệu đặc biệt khác trong Python. Không thường xuyên được sử dụng để đại diện cho không có giá trị.
- Ví dụ,

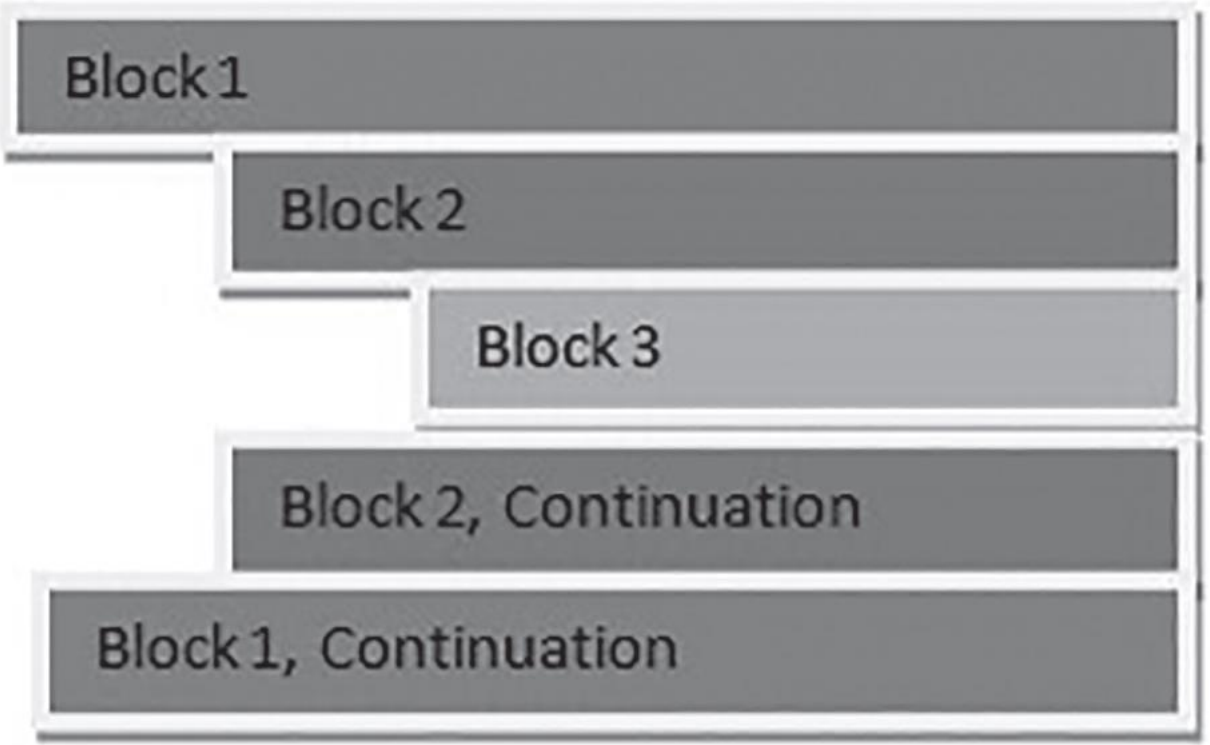
```
>>> money = None
```

(Không có giá trị nào được gán cho tiền khả biến).

Canh lề

- Trong Python, các chương trình được cấu trúc thông qua thụt đầu dòng. Nguyên tắc này làm cho mã trông rõ ràng hơn và dễ hiểu và dễ đọc hơn.
- Các câu lệnh được viết dưới một câu lệnh khác với cùng một thụt lề được hiểu là thuộc cùng một khối mã.
- Nếu có một câu lệnh tiếp theo với ít thụt lề hơn về bên trái, thì nó chỉ có nghĩa là phần cuối của khối mã trước đó.

Canh lề


Block 1
Block 2
Block 3
Block 2, Continuation
Block 1, Continuation

Chú thích

- Chú thích là một phần quan trọng của bất kỳ chương trình nào.
- Chú thích là một văn bản mô tả những gì chương trình đang cố gắng thực hiện và bị bỏ qua bởi trình thông dịch Python.
- Chú thích được sử dụng để giúp bạn và các lập trình viên khác hiểu, duy trì và gỡ lỗi chương trình.
- Python sử dụng hai loại chú thích: chú thích một dòng và chú thích nhiều dòng.

Chú thích

- Chú thích dòng đơn: Sử dụng ký hiệu (#) để bắt đầu viết chú thích. Biểu tượng (#) tạo nên tất cả văn bản theo sau nó trên cùng một dòng vào một chú thích.
- Chú thích nhiều dòng: Sử dụng dấu ngoặc kép, "" hoặc """. Những dấu ngoặc kép này thường được sử dụng cho các chuỗi nhiều dòng.

Đọc đầu vào

- Trong Python, hàm input() được sử dụng để thu thập dữ liệu từ người dùng. Cú pháp cho hàm đầu vào là:

variable_name = input([prompt])

- Lời nhắc (prompt) là một chuỗi được viết bên trong dấu ngoặc đơn được in trên màn hình. Lời nhắc câu lệnh cung cấp một dấu hiệu cho người dùng về giá trị cần được nhập thông qua bàn phím.
- Nhấn phím Enter, chương trình sẽ tiếp tục và đầu vào trả về những gì người dùng được gõ dưới dạng một chuỗi.

Đọc đầu vào

```
1. >>> person = input("What is your name?")
2. What is your name? Carrey
3. >>> person
   'Carrey'
```

In đầu ra

- Hàm print() cho phép một chương trình hiển thị văn bản trên bảng điều khiển.
- Chức năng in sẽ in mọi thứ dưới dạng chuỗi và mọi thứ chưa phải là chuỗi sẽ tự động được chuyển đổi thành biểu diễn chuỗi của nó.
- Ví dụ, >>> print("Hello World!!")
  Hello World!!

In đầu ra

- Phương thức str.format(): Sử dụng phương thức format() nếu bạn cần chèn giá trị của một biến, biểu thức hoặc một đối tượng thành một chuỗi khác và hiển thị nó cho người dùng dưới dạng một chuỗi duy nhất.
- Phương thức format() trả về một chuỗi mới với các giá trị được chèn vào. Phương thức format() sử dụng các đối số của nó để thay thế một giá trị thích hợp cho mỗi định dạng mã trong mẫu.

str.format(p0, p1, ..., k0=v0, k1=v1, ...)

In đầu ra

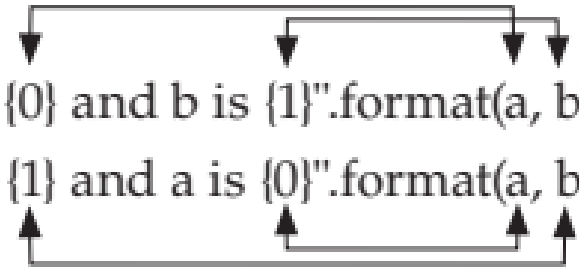
```
1. country = input("Which country do you live in?")
2. print("I live in {0}".format(country))
```

Output

```
Which country do you live in? India
I live in India
```

In đầu ra

```
1. a = 10
2. b = 20
3. print("The values of a is {0} and b is {1}".format(a, b))
4. print("The values of b is {1} and a is {0}".format(a, b))
```



OUTPUT

```
The values of a is 10 and b is 20
The values of b is 20 and a is 10
```

In đầu ra

- f-strings là một chuỗi ký tự có tiền tố là “f”. Các chuỗi này có thể chứa các trường thay thế, là các biểu thức được đặt trong dấu ngoặc nhọn {}.
- Các biểu thức được thay thế bằng các giá trị của chúng. Bạn cần chỉ định tên của biến bên trong dấu ngoặc nhọn để hiển thị giá trị của nó.
- Một f ở đầu chuỗi cho Python biết để cho phép bất kỳ tên biến hợp lệ trong chuỗi.

In đầu ra

```
1. country = input("Which country do you live in?")
2. print(f"I live in {country}")
```

OUTPUT

```
Which country do you live in? India
I live in India
```

In đầu ra

```
1. radius = int(input("Enter the radius of a circle"))
2. area_of_a_circle = 3.1415 * radius * radius
3. circumference_of_a_circle = 2 * 3.1415 * radius
4. print(f"Area={area_of_a_circle}andCircumference={circumference_of_a_circle}")
```

OUTPUT

```
Enter the radius of a circle 5
Area = 78.53750000000001 and Circumference = 31.415000000000003
```

In đầu ra

```
1. number_of_days = int(input("Enter number of days"))
2. number_of_years = int(number_of_days/365)
3. number_of_weeks = int(number_of_days % 365 / 7)
4. remaining_number_of_days = int(number_of_days % 365 % 7)
5. print(f"Years = {number_of_years}, Weeks = {number_of_weeks}, Days =
   {remaining_number_of_days}")
```

OUTPUT

```
Enter number of days375
Years = 1, Weeks = 1, Days = 3
```

Chuyển đổi kiểu dữ liệu

- Bạn có thể ép kiểu hoặc chuyển đổi một cách rõ ràng một biến từ kiểu này sang kiểu khác.
- Hàm int(): Để chuyển đổi một số thực hoặc một chuỗi thành một số nguyên.
- Hàm float(): Trả về một số dấu phẩy động được xây dựng từ một số hoặc chuỗi.
- Hàm str(): Trả về một chuỗi mà ta có thể đọc được.

Chuyển đổi kiểu dữ liệu

- Hàm chr(): Chuyển đổi một số nguyên thành một chuỗi một ký tự có mã ASCII giống với số nguyên. Giá trị số nguyên phải nằm trong khoảng 0–255.
- Hàm complex(): Để in một số phức với giá trị real+image*j hoặc chuyển đổi một chuỗi hoặc một số thành một số phức. Nếu phần ảo bị bỏ qua, nó sẽ mặc định bằng 0 và nếu cả hai đối số là bị bỏ qua, hàm complex () trả về 0j.

Chuyển đổi kiểu dữ liệu

- Hàm ord(): Trả về một số nguyên đại diện cho điểm mã Unicode cho ký tự Unicode.
- Hàm hex(): Chuyển đổi một số nguyên (có kích thước bất kỳ) thành một chuỗi thập lục phân viết thường có tiền tố là “0x”.
- Hàm oct(): Chuyển đổi một số nguyên (có kích thước bất kỳ) thành một chuỗi bát phân có tiền tố là “0o”.

Chuyển đổi kiểu dữ liệu

```
1. float_to_int = int(3.5)
2. string_to_int = int("1") #number treated as string
3. print(f"After Float to Integer Casting the result is {float_to_int}")
4. print(f"After String to Integer Casting the result is {string_to_int}")
```

OUTPUT

```
After Float to Integer Casting the result is 3
After String to Integer Casting the result is 1
```

Chuyển đổi kiểu dữ liệu

1. >>>numerical_value = input("Enter a number")
   Enter a number 9
2. >>> numerical_value
   '9'
3. >>> numerical_value = int(input("Enter a number"))
   Enter a number 9
4. >>> numerical_value
   9

Chuyển đổi kiểu dữ liệu

```
1. int_to_float = float(4)
2. string_to_float = float("1") #number treated as string
3. print(f"After Integer to Float Casting the result is {int_to_float}")
4. print(f"After String to Float Casting the result is {string_to_float}")
```

OUTPUT

```
After Integer to Float Casting the result is 4.0
After String to Float Casting the result is 1.0
```

Chuyển đổi kiểu dữ liệu

```
1. int_to_string = str(8)
2. float_to_string = str(3.5)
3. print(f"After Integer to String Casting the result is {int_to_string}")
4. print(f"After Float to String Casting the result is {float_to_string}")
```

OUTPUT

```
After Integer to String Casting the result is 8
After Float to String Casting the result is 3.5
```

Chuyển đổi kiểu dữ liệu

```
1. ascii_to_char = chr(100)
2. print(f'Equivalent Character for ASCII value of 100 is {ascii_to_char}')
```

OUTPUT

```
Equivalent Character for ASCII value of 100 is d
```

Chuyển đổi kiểu dữ liệu

```
1. complex_with_string = complex("1")
2. complex_with_number = complex(5, 8)
3. print(f"Result after using string in real part {complex_with_string}")
4. print(f"Result after using numbers in real and imaginary part {complex_with_
   number}")
```

OUTPUT

```
Result after using string in real part (1+0j)
Result after using numbers in real and imaginary part (5+8j)
```

Chuyển đổi kiểu dữ liệu

```
unicode_for_integer = ord('4')
unicode_for_alphabet = ord("Z")
unicode_for_character = ord("#")
print(f"Unicode code point for integer value of 4 is {unicode_for_integer}")
print(f"Unicode code point for alphabet 'Z' is {unicode_for_alphabet}")
print(f"Unicode code point for character '#' is {unicode_for_character}")
```

```
Unicode code point for integer value of 4 is 52
Unicode code point for alphabet 'Z' is 90
Unicode code point for character '#' is 35
```

Chuyển đổi kiểu dữ liệu

```
1. int_to_hex = hex(255)
2. print(f"After Integer to Hex Casting the result is {int_to_hex}")
```

OUTPUT

```
After Integer to Hex Casting the result is 0xff
```

```
1. int_to_oct = oct(255)
2. print(f"After Integer to Hex Casting the result is {int_to_oct}")
```

OUTPUT

```
After Integer to Hex Casting the result is 0o377
```

Hàm type()

- Cú pháp cho hàm type() là:

type(object)

- Hàm type() trả về kiểu dữ liệu của đối tượng đã cho. Toán tử được đánh giá là True nếu các giá trị của toán hạng ở hai bên của toán tử trỏ đến cùng một đối tượng và False ngược lại.

Hàm type()

```
1. >>> type(1)
   <class 'int'>
2. >>> type(6.4)
   <class 'float'>
3. >>> type("A")
   <class 'str'>
4. >>> type(True)
   <class 'bool'>
```

Hàm type()

```
1. >>> x = "Seattle"
2. >>> y = "Seattle"
3. >>> x is y
   True
```