CHƯƠNG 1. GIỚI THIỆU

1.1. Tạo và thực thi một chương trình Python

Mã Python có thể được viết bằng bất kỳ trình soạn thảo văn bản thuần túy nào có thể tải và lưu văn bản bằng cách sử dụng bảng mã ký tự ASCII hoặc UTF-8 Unicode. Mặc định, các tệp Python được sử dụng mã hóa ký tự UTF-8. Các tệp Python thường có phần mở rộng là `.py`, và các chương trình Python GUI (Giao diện Người dùng Đồ họa) thường có phần mở rộng là `.pyw`, đặc biệt là trên Windows và Mac OS X. Trong tài liệu này, tôi luôn sử dụng phần mở rộng của `.py` cho các chương trình console và module, và .pyw cho các chương trình GUI.

Chỉ để đảm bảo rằng mọi thứ được thiết lập chính xác và để hiển thị ví dụ cổ điển đầu tiên, hãy tạo một tệp có tên `hello.py` trong trình soạn văn bản thuần túy (Windows Notepad), với nội dung sau:

```
#!/usr/bin/env python3

print("Hello", "World!")
```

Dòng đầu tiên là chú thích, trong Python chú thích bắt đầu bằng dấu `#`. Dòng thứ hai là dòng trống, Python sẽ bỏ qua các dòng trống trong chương trình. Dòng thứ ba là mã lệnh Python, ở ví dụ trên là gọi hàm `print` với hai tham số, mỗi tham số là một kiểu chuỗi.

Mỗi câu lệnh xuất hiện trong file `.py` sẽ được thực hiện tuần tự, bắt đầu từ câu lệnh đầu tiên đến câu lệnh cuối cùng.

Giả sử lưu chương trình Python trong thư mục C:\Py3eg, với tên file `hello.py` và đóng chương trình soạn thảo. Để thực thi chương trình, ta có thể sử dụng chương trình trình thông dịch Python và thông thường điều này được thực hiện bên trong cửa sổ lệnh (Command Prompt). Khởi động cửa sổ lệnh và nhập phần:

```
C:\>cd c:\py3eg
C:\py3eg\>c:\python31\python.exe hello.py
```

Kết quả xuất hiện trên màn hình:

```
Hello World!
```

1.2. Các điểm nổi bật của Python

1.2.1. Kiểu dữ liệu

Một điều cơ bản mà bất kỳ ngôn ngữ lập trình nào cũng phải có đó là biểu diễn dữ liệu. Python cung cấp một số kiểu dữ liệu tích hợp sẵn, nhưng chúng ta sẽ tìm hiểu hai kiểu cơ bản sau. Python biểu diễn các số nguyên (số nguyên dương và âm) bằng cách sử dụng kiểu `int` và biểu diễn các chuỗi (chuỗi ký tự Unicode) sử dụng kiểu `str`. Ví dụ:

```
-973
21062458333711437339583605536734086463779019080109822250862195507 20
"Infinitely Demanding"
'Simon Critchley'
'positively αβγ ÷©'
''
```

Số thứ hai được hiển thị là 2217, kích thước số nguyên của Python chỉ bị giới hạn bởi bộ nhớ máy, không phải bởi một số byte cố định. Các chuỗi có thể được xác định bằng dấu ngoặc kép hoặc đơn, miễn là cùng một loại được sử dụng ở cả hai đầu và vì Python sử dụng Unicode, các chuỗi không bị giới hạn ở các ký tự ASCII, như chuỗi gần cuối thể hiện.

Python sử dụng dấu ngoặc vuông (`[]`) để truy cập một kí tự từ một chuỗi. Ví dụ: nếu chúng ta đang ở trong Python Shell (trong trình thông dịch tương tác hoặc trong IDLE), chúng ta có thể nhập như sau:

```
>>> "Hard Times"[5]
'T'
>>> "giraffe"[0]
'g'
```

Trong Python, cả `str` và các kiểu số cơ bản như `int` đều không thay đổi được, nghĩa là, sau khi được đặt, giá trị của chúng không thể thay đổi, chúng ta không thể sử dụng chúng để đặt một ký tự mới.

Để chuyển đổi dữ liệu từ kiểu này sang kiểu khác, chúng ta có thể sử dụng cú pháp `datatype(item)`. Ví dụ:

```
>>> int("45")
45
>>> str(912)
```

```
'912'
```

Chuyển đổi `int()` chấp nhận khoảng trắng ở đầu và cuối, vì vậy int (" 45 ") cũng sẽ hoạt động tốt. Chuyển đổi `str()` có thể được áp dụng cho hầu hết mọi mục dữ liệu. Nếu một chuyển đổi không thành công, Python sẽ phát sinh ngoại lệ.

1.2.2. Tham chiếu đối tượng

Khi có một số kiểu dữ liệu, thì cần có các biến để lưu trữ chúng. Python không có các biến như vậy, nhưng thay vào đó có các tham chiếu đối tượng.

Xét ví dụ đơn giản sau:

```
x = "blue"
y = "green"
z = x
```

Cú pháp đơn giản là `objectReference = value`. Không cần khai báo trước và không cần chỉ định loại của giá trị. Khi Python thực thi câu lệnh đầu tiên, nó tạo một đối tượng `str` với nội dung là `"blue"` và tạo một tham chiếu đối tượng được gọi là `x` tham chiếu đến đối tượng `str`. Chúng ta có thể nói rằng biến `x` đã được gán là chuỗi `"blue"`. Câu lệnh thứ hai cũng tương tự. Câu lệnh thứ ba tạo một tham chiếu đối tượng mới được gọi là `z` và đặt nó tham chiếu đến cùng một đối tượng mà tham chiếu đối tượng `x` tham chiếu đến (trong trường hợp này là str chứa văn bản `"blue"`).

Tiếp tục với ví dụ `x`, `y`, `z`:

```
print(x, y, z) # in ra: blue green blue
z = y
print(x, y, z) # in ra: blue green green
x = z
print(x, y, z) # in ra: green green green
```

Sau câu lệnh thứ tư (`x = z`), cả ba tham chiếu đối tượng đều tham chiếu đến cùng một `str`. Vì không còn tham chiếu đối tượng nào nữa đến chuỗi `"blue"`, Python có thể giải phóng bộ nhớ cho nó.

CHƯƠNG 2. KIỂU DỮ LIỆU VÀ THAO TÁC NHẬP/XUẤT

2.1. Định danh và từ khóa

Định danh trong Python là chuỗi kí tự khác rỗng có độ dài bất kì bắt đầu bởi một chữ cái sau đó có thể là chữ cái hoặc chữ số, dấu gạch dưới, … các kí tự trong bộ mã unicode. Định danh có phân biệt chữ hoa, chữ thường. Định danh không được trùng với các từ khóa sau:

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

2.2. Số nguyên

2.2.1. Kiểu int

Kích thước của một số nguyên chỉ bị giới hạn bởi bộ nhớ của máy, vì vậy, có thể dễ dàng tạo và làm việc với các số nguyên dài hàng trăm chữ số, mặc dù chúng sẽ chậm sử dụng hơn so với các số nguyên truyền thống.

Bảng 2.1. *Các phép toán và hàm trên kiểu số.*

Cú pháp	Mô tả
`x + y`	Cộng số x và số y
`x - y`	Trừ số x cho y
`x * y`	x nhân y
`x / y`	Chia x cho y; sinh ra số thực hoặc số phức (nếu x hoặc y là số phức)
`x // y`	Chia x cho y, làm tròn phần thập phân vì vậy kết quả luôn là số nguyên
`x % y`	Chia lấy phần dư x cho y
`x ** y`	x mũ y
`-x`	Đổi dấu
`+x`	Không làm gì
`abs(x)`	Trị tuyệt đối của x

`divmod(x, y)`	Trả về thương số và phần dư của phép chia x cho y (một bộ 2 số nguyên)
`pow(x, y)`	Tính x mũ y (giống phép toán **)
`pow(x, y, z)`	Giống như (x ** y) % z
`round(x, n)`	Trả về số x đã làm tròn đến n chữ số. n âm làm tròn sau dấu thập phân, n dương làm tròn trước dấu thập phân. Giá trị trả về cùng kiểu với x

Bảng 2.2. *Hàm chuyển đổi cơ số.*

Cú pháp	Mô tả
`bin(i)`	Trả về biểu diễn nhị phân của số nguyên i (dạng chuỗi str)
`hex(i)`	Trả về biểu diễn thập lục phân của số nguyên i (dạng chuỗi)
`int(x)`	Chuyển một đối tượng x sang số nguyên.
`int(s, base)`	Chuyển một chuỗi s sang số nguyên với cơ số base (2-36)
`oct(i)`	Trả về biểu diễn bát phân của số nguyên i (dạng chuỗi)

Tất cả các phép toán hai ngôi (+, -, /, //, % và **) đều có phiên bản gán tăng cường (+=, -=, /=, //=, %= và **=) trong đó `x op= y` tương đương với `x = x op y`.

2.2.2. Kiểu bool

Python phiên bản mới có định nghĩa kiểu `bool` với hai giá trị `True` và `False`. Tuy nhiên những phiên bản trước đó coi giá trị 0 là `False` và khác 0 là `True`. Kiểu logic cũng có 3 phép toán cơ bản: `and`, `or`, `not`.

2.3. Số thực

Python cung cấp ba loại giá trị dấu phẩy động: loại `float`, `complex` và `Decimal`.

2.3.1. Kiểu số thực dấu phẩy động (float)

Tất cả các toán tử và hàm số trong Bảng 2.1 có thể được sử dụng với `float`, bao gồm cả các phiên bản phép gán tăng cường.

Bảng 2.3. *Các hàm toán học kiểu số trong module math.*

Cú pháp	Mô tả
`math.acos(x)`	Trả về arc cosine của x (radians)
`math.acosh(x)`	Trả về arc hyperbolic cosine của x (radians)
`math.asin(x)`	Trả arc sine của x (radians)
`math.asinh(x)`	Trả về arc hyperbolic sine của x (radians)
`math.atan(x)`	Trả về arc tangent của x (radians)
`math.atan2(y, x)`	Trả về arc tangent của y / x (radians)
`math.atanh(x)`	Trả về arc hyperbolic tangent của x (radians)
`math.ceil(x)`	Trả về số nguyên nhỏ nhất lớn hơn hoặc bằng x; ví dụ: math.ceil(5.4) == 6

Cú pháp	Mô tả
`math.copysign(x,y)`	Trả về số x với dấu của y
`math.cos(x)`	Trả về cosine của x (radians)
`math.cosh(x)`	Trả về hyperbolic cosine của x (radians)
`math.degrees(r)`	Đổi số thực r từ radians sang độ
`math.e`	Hằng số e; xấp xĩ 2.7182818284590451
`math.exp(x)`	Trả về ex, i.e., math.e ** x
`math.fabs(x)`	Trả về trị tuyệt đối x i.e. (float)
`math.factorial(x)`	Trả về x!
`math.floor(x)`	Trả về số nguyên lớn nhất nhỏ hơn hoặc bằng x; ví dụ: math.floor(5.4) == 5
`math.fmod(x, y)`	Trả về phần dư (số thực) của phép chia x/y
`math.frexp(x)`	Trả về hai giá trịReturns a 2-tuple with the mantissa (as a float) and the exponent (as an int) so, x = m × 2e; see math.ldexp()
`math.fsum(i)`	Trả về tổng các giá trị trong danh sách i (float)
`math.hypot(x, y)`	$\sqrt{x^2 + y^2}$
`math.isinf(x)`	Trả về True nếu số thực x là ± ∞
`math.isnan(x)`	Trả về True nếu số thực x là không phải số (“not a number”)
`math.ldexp(m, e)`	Trả về m × 2e; ngược với hàm math.frexp()
`math.log(x, b)`	Trả về logbx; b tùy chọn (ngầm định là số e)
`math.log10(x)`	Trả về log10x
`math.log1p(x)`	Trả về loge(1+ x)
`math.modf(x)`	Trả về phần thực và nguyên của số thực x
`math.pi`	Hằng số π; xấp xĩ 3.1415926535897931
`math.pow(x, y)`	Trả về xy (float)
`math.radians(d)`	Chuyển số thực d từ độ sang radians
`math.sin(x)`	Trả về sine của x (radians)
`math.sinh(x)`	Trả về hyperbolic sine của x (radians)
`math.sqrt(x)`	Trả về $\sqrt{x}$
`math.tan(x)`	Trả về tangent của x (radians)
`math.tanh(x)`	Trả về hyperbolic tangent của x (radians)
`math.trunc(x)`	Trả về phần nguyên của x

Để sử dụng các hàm này, yêu cầu phải: `import math`.

2.3.2. Kiểu số phức (complex)

Kiểu dữ liệu số phức là kiểu bất biến chứa một cặp số thực, một phần biểu diễn phần thực và phần còn lại là phần ảo của một số phức. Hằng số phức được viết với phần thực và phần ảo được nối bằng dấu + hoặc – và với phần ảo theo sau là kí hiệu `j`. Dưới đây là một số ví dụ: `3.5 + 2j`, `0.5j, 4 + 0j`, `–1–3.7j`. Lưu ý rằng nếu phần thực là

0, chúng ta có thể bỏ qua. Mỗi đối tượng số phức luôn có hai thuộc tính là `real` và `imag` để lưu phần thực và ảo.

Ngoại trừ phép toán `//`, `%`, `divmod()` và hàm ba đối số `pow()`, tất cả các phép toán và hàm số trong số thực `float` đều có thể được sử dụng với các số phức và các phiên bản gán tăng cường cũng vậy. Ngoài ra, số phức có một phương thức `conjugate()` đổi dấu phần ảo.

Để sử dụng các hàm trong kiểu số phức, ta sử dụng: `import cmath`.

2.3.3. Số thập phân (decimal.Decimal)

Trong thực tế, số thực `float` có độ chính xác kém (số chữ số sau dấu thập phân ít, khoảng 16 chữ số). Khi cần số thực có độ chính xác cao hơn, ta sử dụng module `decimal` để tạo ra đối tượng `Decimal`.

Để tạo đối tượng `Decimal`, ta thực hiện các câu lệnh sau:

```
>>> import decimal
>>> a = decimal.Decimal(9876)
>>> b = decimal.Decimal("54321.012345678987654321")
>>> a + b
Decimal('64197.012345678987654321')
```

Số thập phân được tạo bằng cách sử dụng hàm `decimal.Decimal()`. Hàm này có thể nhận một số nguyên hoặc một đối số chuỗi, nhưng không phải là một số `float`. Nếu một chuỗi được sử dụng, nó có thể sử dụng ký hiệu thập phân đơn giản hoặc ký hiệu hàm mũ. Ngoài việc cung cấp độ chính xác, sự biểu diễn chính xác của số thập phân. Các số thập phân có thể được so sánh chính xác hơn.

Tất cả các toán tử và hàm số được liệt kê trong Bảng 2.1, bao gồm cả các phiên bản gán tăng cường, có thể được sử dụng với đối tượng `decimal.Decimal`, nhưng có một số lưu ý. Nếu toán tử ** có toán hạng bên trái `decimal.Decimal` thì toán hạng bên phải của nó phải là số nguyên. Tương tự, nếu đối số đầu tiên của hàm `pow()` là `decimal.Decimal`, thì đối số thứ hai và thứ ba tùy chọn của nó phải là số nguyên.

2.4. Kiểu chuỗi (str)

Các chuỗi được biểu diễn bằng kiểu dữ liệu `str` chứa một chuỗi các ký tự Unicode. Kiểu dữ liệu `str` có thể được gọi như một hàm để tạo các đối tượng chuỗi, không có đối số, nó trả về một chuỗi rỗng, với đối số không phải chuỗi (`nonstring`), nó trả về dạng chuỗi của đối số và với đối số chuỗi, nó trả về một bản sao của chuỗi. Hàm `str()` cũng có thể được sử dụng như một hàm chuyển đổi, trong trường hợp đó, đối số đầu tiên phải

là một chuỗi hoặc một cái gì đó có thể chuyển đổi thành một chuỗi.

Các chuỗi kí tự được tạo bằng cách sử dụng dấu nháy kép hoặc dấu nháy đơn miễn là giống nhau ở cả hai đầu. Ngoài ra, chúng ta có thể sử dụng một chuỗi ba dấu nháy kép. Ví dụ:

```
text = """A triple quoted string like this can include 'quotes' and
"quotes" without formality. We can also escape newlines \
so this particular string is actually only two lines long."""
```

Bảng 2.4. *Các kí tự đặc biệt trong Python.*

Kí tự	Ý nghĩa
\newline	Tạo dòng mới
\\	Dấu Backslash (\)
\'	Dấu nháy đơn (’)
\"	Dấu nháy kép (")
\a	ASCII bell (BEL)
\b	ASCII backspace (BS)
\f	ASCII formfeed (FF)
\n	ASCII linefeed (LF)
\N{name}	Kí tự Unicode với tên
\ooo	Kí tự với giá trị số ở hệ 8
\r	ASCII carriage return (CR)
\t	ASCII tab (TAB)
\uhhhh	Kí tự Unicode với giá trị hệ 16 độ dài 16-bit
\Uhhhhhhhh	Kí tự Unicode với giá trị hệ 16 độ dài 32-bit
\v	ASCII vertical tab (VT)
\xhh	Kí tự với giá trị hệ 16 độ dài 8-bit

Nếu chúng ta muốn viết một chuỗi ký tự dài trải dài trên hai hoặc nhiều dòng nhưng không sử dụng chuỗi ba dấu ngoặc kép, chúng ta có thể thực hiện một số cách sau:

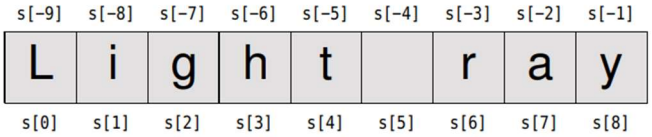
```
t = "This is not the best way to join two long strings " + \
"together since it relies on ugly newline escaping"
s = ("This is the nice way to join two long strings "
"together; it relies on string literal concatenation.")
```

Vì các tệp .py được mặc định sử dụng bảng mã UTF-8 Unicode, chúng ta có thể viết bất kỳ ký tự Unicode nào trong chuỗi ký tự bằng cách sử dụng kí tự \ như sau:

```
>>> euros = "€ \N{euro sign} \u20AC \U000020AC"
>>> print(euros)
>>> € € € €
```

2.4.2. Cắt chuỗi

Chuỗi trong Python được đánh số từ 0 đến hết chuỗi. Nhưng cũng có thể sử dụng chỉ số âm với kí tự chuỗi cùng (bên phải) có chỉ số là `-1` đến kí tự đầu tiên. Ví dụ `s="Light ray"`.



Để cắt chuỗi ta có 3 cách sau:

```
seq[start]
seq[start:end]
seq[start:end:step]
```

Trong đó `start` vị trí bắt đầu, `end` vị trí kết thúc, `step` bước nhảy. Nếu bước nhảy là số âm thì cắt chuỗi từ phải qua trái.

2.4.3. Phương thức và phép toán trên chuỗi

Trên chuỗi có các phép toán: `in` (kiểm tra xuất hiện), `+` (ghép chuỗi), `+=` (ghép vào cuối), `*` (sao chép) và `*=` (sao chép tăng cường).

Bảng 2.5. *Các phương thức trên chuỗi.*

Tên	Ý nghĩa
`s.capitalize()`	Trả về bản sao của chuỗi s với kí tự đầu là chữ hoa.
`s.center(width, char)`	Trả về bản sao của s được căn giữa trong một chuỗi có chiều dài chiều rộng được đệm bằng dấu cách hoặc tùy chọn bằng char (chuỗi có độ dài 1)
`s.count(t, start, end)`	Trả về số lần xuất hiện của chuỗi t trong chuỗi s (hoặc trong phạm vi start:end của s)
`s.encode(encoding, err)`	Trả về một đối tượng byte biểu diễn cho chuỗi bằng cách sử dụng mã hóa mặc định hoặc sử dụng mã hóa được chỉ định và xử lý lỗi theo đối số err tùy chọn
`s.endswith(x, start, end)`	Trả về True nếu s (hoặc đoạn start:end của s) kết thúc bằng str x, ngược lại trả về False.
`s.expandtabs(size)`	Trả về bản sao của s với các tab được thay thế bằng dấu cách theo bội số của 8 hoặc kích thước nếu được chỉ định
`s.find(t, start, end)`	Trả về vị trí tận cùng bên trái của t trong s (hoặc trong phần start: end của s) hoặc -1 nếu không tìm thấy. Sử dụng `str.rfind()` để tìm vị trí ngoài cùng bên phải.

Tên	Ý nghĩa
`s.format(...)`	Trả về bản sao của s được định dạng theo các đối số đã cho.
`s.index(t, start, end)`	Trả về vị trí tận cùng bên trái của t trong s (hoặc trong phần bắt đầu: kết thúc của s) hoặc phát sinh ValueError nếu không tìm thấy. Sử dụng str.rindex() để tìm từ bên phải
`s.isalnum()`	Trả về True nếu s không rỗng và mọi ký tự trong s đều là chữ và số
`s.isalpha()`	Trả về True nếu s không rỗng và mọi ký tự trong s đều là chữ cái
`s.isdecimal()`	Trả về True nếu s không rỗng và mọi ký tự trong s là chữ số hệ 10 (Unicode)
`s.isdigit()`	Trả về True nếu s không rỗng và mọi ký tự trong s là một chữ số (ASCII)
`s.isidentifier()`	Trả về True nếu s không rỗng và là một định danh hợp lệ
`s.islower()`	Trả về True nếu s có ít nhất một ký tự có thể viết thường và tất cả các ký tự có thể viết thường của nó đều là ký tự viết thường.
`s.isnumeric()`	Trả về True nếu s không rỗng và mọi ký tự trong s là ký tự (Unicode) số như: chữ số hoặc phân số
`s.isprintable()`	Trả về True nếu s rỗng hoặc nếu mọi ký tự trong s được coi là có thể in được, bao gồm khoảng trắng, nhưng không phải dòng mới.
`s.isspace()`	Trả về True nếu s không rỗng và mọi ký tự trong s là khoảng trắng
`s.istitle()`	Trả về True nếu s là một chuỗi tiêu đề không rỗng
`s.isupper()`	Trả về True nếu chuỗi s có ít nhất một ký tự có thể viết hoa và tất cả các ký tự có thể viết hoa của nó đều là chữ hoa;
`s.join(seq)`	Trả về phần nối mọi mục trong tập hợp seq.
`s.ljust(width, char)`	Trả về bản sao của s được căn trái trong chuỗi có độ dài width được đệm bằng dấu cách hoặc tùy chọn bằng char (chuỗi có độ dài 1).
`s.lower()`	Trả về bản sao viết thường của s;
`s.maketrans()`	Companion of str.translate(); see text for details
`s.partition(t)`	Trả về một bộ ba phần, phần của s trước t, phần chuỗi t, phần của s sau t.
`s.replace(t, u, n)`	Trả về bản sao của s với mọi (hoặc tối đa n nếu cho trước) lần xuất hiện của chuỗi t được thay thế bằng chuỗi u

Tên	Ý nghĩa
`s.split(t, n)`	Trả về danh sách các chuỗi phân tách nhiều nhất n lần dựa trên chuỗi t; nếu không được đưa ra chuỗi t, phân tách dựa trên khoảng trắng.
`s.splitlines(f)`	Trả về danh sách các dòng được tạo ra bằng cách tách s trên các đầu cuối dòng, loại bỏ các đầu cuối trừ khi f là True
`s.startswith(x, start, end)`	Trả về True nếu s (hoặc phần start:end của s) bắt đầu bằng chuỗi x hoặc với bất kỳ chuỗi nào trong bộ tuple x;
`s.strip(chars)`	Trả về bản sao của s với khoảng trắng ở đầu và cuối (hoặc các ký tự trong char) bị xóa; str.lstrip() chỉ xóa ở đầu và str.rstrip() ở cuối
`s.swapcase()`	Trả về bản sao của s với các ký tự viết hoa sẽ viết thường và các ký tự viết thường được viết hoa;
`s.title()`	Trả về bản sao của s trong đó chữ cái đầu tiên của mỗi từ là chữ hoa và tất cả các chữ cái khác là chữ thường.
`s.translate()`	Companion of str.maketrans(); see text for details
`s.upper()`	Trả về bản sao viết hoa của s;
`s.zfill(w)`	Trả về một bản sao của s, nếu ngắn hơn w sẽ được đệm bằng các số 0 ở đầu để làm cho nó dài w

2.4.4. Định dạng chuỗi bằng phương thức str.format()

Phương thức `str.format()` trả về một chuỗi mới với các trường thay thế trong chuỗi được thay thế bằng các đối số được định dạng phù hợp. Ví dụ:

```
>>> "The novel '{0}' was published in {1}".format("Hard Times",
1854)
"The novel 'Hard Times' was published in 1854"
```

Mỗi trường thay thế được xác định bằng một tên trường trong dấu ngoặc nhọn. Nếu tên trường là một số nguyên, nó được coi là vị trí chỉ mục của một trong các đối số được truyền đến `str.format()`.Vì vậy, trong trường hợp này, trường có tên 0 được thay thế bằng đối số đầu tiên và trường có tên 1 được thay thế bằng đối số thứ hai. Trường thay thế có cú pháp sau đây:

```
{field_name}
{field_name!conversion}
{field_name:format_specification}
{field_name!conversion:format_specification}
```

2.4.5. Định dạng chuỗi bằng kí hiệu %

Cú pháp: `"…..%c.....%d...." % (values)`. Trong đó `values` là giá trị cần định dạng, các kí tự định dạng kiểu dữ liệu sau dấu `%` như sau:

Kí tự	Ý nghĩa
`s`	Chuỗi kí tự
`c`	Kí tự hoặc int
`d`	Số thập phân
`i`	Số nguyên
`u`	Số nguyên không dấu
`o`	Số nguyên không dấu hệ 8
`x`	Số nguyên không dấu hệ 16
`X`	Giống x nhưng kí tự hiển thị chữ hoa
`e`	Số thực dấu phẩy động
`E`	Giống e, nhưng kí tự hoa
`f`	Số thực dấu phẩu động (decimal)
`F`	Giống f, nhưng kí tự chữ hoa
`%`	Hiện dấu %

2.5. Nhập xuất dữ liệu

2.5.1. Nhập dữ liệu từ bàn phím hàm input

Để nhập dữ liệu từ bàn phím, ta sử dụng hàm `input` với cú pháp như sau:

```
Name = input("Dòng thông báo")
```

Khi hàm thực hiện, nó hiện ra một dòng thông báo và chờ người dùng nhập vào một chuỗi dữ liệu và trả về chuỗi dữ liệu đó. Đề nhận được kiểu dữ liệu mong muốn trong lúc nhập, ta phải chuyển đổi kiểu dữ liệu bằng các hàm tương ứng ví dụ như: `int`, `float`, `complex`, `decimal.Decimal`, …

2.5.2. In dữ liệu ra màn hình

Ta có thể in dữ liệu ra màn hình bằng hàm `print` với cú pháp như sau:

```
print([object, ...][, sep=' '][, end='\n'][, file=sys.stdout])
```

Trong đó `object` là đối tượng cần in, `sep` là kí tự ngăn cách giữa các đối tượng (ngầm định là dấu cách), `end` là kí tự xuất hiện ở cuối dòng in (ngầm định là dấu xuống

dòng), `file` chỉ ra đối tượng file để in giá trị đến (ngầm định là màn hình).

2.6. Ví dụ

Ví dụ 2.1. *Phương trình bậc hai là phương trình có dạng ax2 + bx + c = 0 trong đó a ≠ 0. Nghiệm của các phương trình được tính từ công thức* $x = \frac{-b^2 \pm \sqrt{b^2-4ac}}{2a}$. *Trong đó Δ=b2-4ac nếu dương có 2 nghiệm thực, nếu bằng 0 có 1 nghiệm thực và nếu nhỏ hơn không có 2 nghiệm phức. Viết chương trình nhập ba số thực a, b, c, tìm nghiệm của phương trình.*

```
import math
import cmath
a = float(input('Nhập hệ số a: '))
b = float(input('Nhập hệ số b: '))
c = float(input('Nhập hệ số c: '))
d = b**2 - 4*a*c
x1 = None
x2 = None
if d==0:
    x1 = -(b/(2*a))
else:
    if d>0:
        root = math.sqrt(d)
    else:
        root = cmath.sqrt(d)
    x1 = (-b-root)/(2*a)
    x2 = (-b+root)/(2*a)
equation = ("phương trình {0}x\N{SUPERSCRIPT TWO} + {1}x + {2} = 0
có nghiệm x = {3:.3}").format(a, b, c, x1)
if x2 is not None:
    equation += " hoặc x={0:.2}".format(x2)
print(equation)
```

2.7. Bài tập

Bài 2.1. Một quả bóng được ném thẳng đứng trong không khí từ độ cao h0 so với mặt đất với vận tốc ban đầu v0. Độ cao h và vận tốc v tiếp theo của nó được cho bởi các phương trình sau: $h = h_0 + v_0 t - \frac{1}{2} g t^2$, v = v0 – gt, trong đó g=9.8m/s2 là gia tốc trọng trường. Viết đoạn chương trình tìm chiều cao và vận tốc v tại thời điểm t sau khi quả bóng được ném.

Bài 2.2. Viết câu lệnh tính giá trị các biểu thức sau: $a = \frac{2+e^{2.8}}{\sqrt{13}-2}$, $b = \frac{1-(1+ln2)^{-3.5}}{1+\sqrt{5}}$, $c =$

$sin\left(\frac{2-\sqrt{2}}{2+\sqrt{2}}\right)$

Bài 2.3. Viết chương trình nhập vào ba cạnh a, b, c của một tam giác. In ra màn hình chu vi, diện tích và các góc hợp bởi ba cạnh của tam giác đó.

Bài 2.4. Viết chương trình nhập vào số tiền, in ra màn hình số tờ của các mệnh giá 100000, 200000, 500000 tương ứng với số tiền đó.

Bài 2.5. Viết chương trình nhập vào số giây, in ra màn hình giờ:phút:giây tương ứng.