Python String Formatting Methods

Shouke Wei, Ph.D. Professor

Email: shouke.wei@gmail.com

Objective

- % formatting method
- curly braces formatting method
- f-strings formatting method

1. % formatting

- an older method of string formatting that uses the % operator
- the %s marker inserts a string, the %d marker inserts an integer,%f a float

1.1 One variable

In [19]:
```
name = 'Mike'
print('Hello, %s!'% name)
```

```
Hello, Mike!
```

1.2 More than one variable

In [18]:
```
name = 'Jack'
age = 20

print('%s is %d years old.' % (name,age))
```

```
Jack is 20 years old.
```

1.3 A list

In [2]:
```
alist = [5,1,8]
print("A list: %s" % alist)
```

```
A list: [5, 1, 8]
```

1.4 Format the number value

Format float decimal place

- format a float with certain decimal places, such as 0.2, 0.22

In [3]:
```
x = 1
y = 3
z = x/y

print('The result of %d divided by %d is %f.'%(x,y,z))
print('The result of %d divided by %d with one decimal '\
    'place is %.1f.'%(x,y,z))
print('The result of %d divived by %d with '\
    'two decimal places is %.2f.'%(x,y,z))
print('The result of %d divided by %d with '\
    'three decimal places is %.3f.'%(x,y,z))
```

```
The result of 1 divided by 3 is 0.333333.
The result of 1 divided by 3 with one decimal place is 0.3.
The result of 1 divived by 3 with two decimal places is 0.33.
The result of 1 divided by 3 with three decimal places is 0.333.
```

2. Curly brace string formatting

- You can insert more than one value.
- The values can be numbers and other Python objects

2.1 Insert a string and number

In [1]:
```
name = 'Jack'
age = 20

print ('{} is {} years old.'.format(name, age))
```

```
Jack is 20 years old.
```

2.2 Insert a complex data type

- such as list, tuple, ect.

In [5]:
```
alist = [5,1,8]
print("A list: {}.".format(alist))
```

```
A list: [5, 1, 8].
```

2.3 Format the number value

Format float decimal palce

- format a float with certain decimal places, such as 0.2, 0.22

In [10]:
```
x = 1
y = 3
z = x/y

print('The result of {} dived by {} is {}.'.format(x,y,z))
print('The result of {} dived by {} with one decimal'\
    'place is {:.1f}.'.format(x,y,z))
print('The result of {} dived by {} with'\
    'two decimal places is {:.2f}.'.format(x,y,z))
print('The result of {} dived by {} with'\
    'three decimal places is {:.3f}.'.format(x,y,z))
```

```
The result of 1 dived by 3 is 0.3333333333333333.
The result of 1 dived by 3 with one decimalplace is 0.3.
The result of 1 dived by 3 withtwo decimal places is 0.33.
The result of 1 dived by 3 withthree decimal places is 0.333.
```

3. f-string method

- a new method only after Python >= version 3.6
- An `f` prefix at the beginning of the string tells Python to insert any currently valid variables into the string
- The most practical one

3.1 One variable

In [6]:
```
name = 'Jack'

print(f'Hello, {name}.')
```

```
Hello, Jack.
```

3.2 More than one variable

In [74]:
```
name = 'Jack'
age = 20

print(f'{name} is {age} years old.')
```

```
Jack is 20 years old.
```

3.3 f-string List

In [9]:
```
alist = [5,1,8]

print(f"A list: {alist}")
```

```
A list: [5, 1, 8]
```

3.4 Formating floats

In [2]:
```
x = 1
y = 3
z = x/y

print(f'{x} is dived by {x} is {z:.4f}.')
```

```
1 is dived by 1 is 0.3333.
```

3.5 f-string Dictionaries

In [5]:
```
fruit = {
  'name': 'Apple',
  'price': '3.0'
}

print(f"{fruit['name']} is ${fruit['price']}")
```

```
Apple is $3.0
```

3.6 f-string expression

In [12]:
```
apple_amount = 5  # kg
cost = 3.0 # Dollar per kg

print(f'Total cost of the apple is ${apple_amount * cost}.')
```

```
Total cost of the apple is $15.0.
```

3.7 multiline f-string

In [14]:
```
name = 'Jack Smith'
age = 25
occupation = 'Professor'

file = (
    f'Name: {name}\n'
    f'Age: {age}\n'
    f'Occupation: {occupation}'
)
print(file)
```

```
Name: Jack Smith
Age: 25
Occupation: Professor
```

3.8 f-string calling function

In [15]:
```
def additor(x, y):

    return x + y

a = 5
b = 7

print(f'Sum of {a} and {b} is {additor(a, b)}')
```

```
Sum of 5 and 7 is 12
```

3.9 f-string objects

- the objects must have either **str**() or **repr**() magic functions defined

In [16]:
```
class User:
    def __init__(self, name, occupation):
        self.name = name
        self.occupation = occupation

    def __repr__(self):
        return f"{self.name} is a {self.occupation}"

u = User('John Doe', 'gardener')

print(f'{u}')
```

```
John Doe is a gardener
```

3.10 f-string format width

- The value may be filled with spaces or other characters if the value is shorter than the specified width
- The example prints three columns. Each of the columns has a predefined width. The first column uses 0 to fill shorter values.

In [4]:
```
for x in range(1, 11):
    print(f'{x:02} {x*x:3} {x*x*x:4}')
```

```
01   1    1
02   4    8
03   9   27
04  16   64
05  25  125
06  36  216
07  49  343
08  64  512
09  81  729
10 100 1000
```

In [9]:
```python
s1 = 'a'
s2 = 'ab'
s3 = 'abc'
s4 = 'abcd'

print(f'{s1:>10}')
print(f'{s2:>10}')
print(f'{s3:>10}')
print(f'{s4:>10}')
```

```
         a
        ab
       abc
      abcd
```