

ZetCode

[All](#) [Spring Boot](#) [Python](#) [C#](#) [Java](#) [JavaScript](#) [Subscribe](#)

Python f-string

last modified July 6, 2020

Python f-string tutorial shows how to format strings in Python with f-string.

Python f-string

Python f-string is the newest Python syntax to do string formatting. It is available since Python 3.6. Python f-strings provide a faster, more readable, more concise, and less error prone way of formatting strings in Python.

The f-strings have the f prefix and use {} brackets to evaluate values.

←

Ads by Google

Send feedback

Why this ad? ▶

Format specifiers for types, padding, or aligning are specified after the colon character; for instance: f'{price:.3}', where price is a variable name.

Python string formatting

The following example summarizes string formatting options in Python.

formatting_strings.py

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

name = 'Peter'
age = 23

print('%s is %d years old' % (name, age))
print('{} is {} years old'.format(name, age))
print(f'{name} is {age} years old')
```

The example formats a string using two variables.

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

This is the oldest option. It uses the % operator and classic string format specifiers such as %s and %d.

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

Since Python 3.0, the format() function was introduced to provide advance formatting options.

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

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

Python f-strings are available since Python 3.6. The string has the f prefix and uses { } to evaluate variables.

```
$ python formatting_string.py
Peter is 23 years old
Peter is 23 years old
Peter is 23 years old
```

We have the same output.

Python f-string expressions

We can put expressions between the { } brackets.

expressions.py

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

bags = 3
apples_in_bag = 12

print(f'There are total of {bags * apples_in_bag} apples')
```

The example evaluates an expression inside f-string.

Ads by Google

Send feedback

Why this ad? ▶

```
$ python expressions.py
There are total of 36 apples
```

This is the output.

Python f-string dictionaries

We can work with dictionaries in f-strings.

dicts.py

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

```
print(f"{user['name']} is a {user['occupation']}")
```

The example evaluates a dictionary in an f-string.

```
$ python dicts.py
John Doe is a gardener
```

This is the output.

Python f-string debug

Python 3.8 introduced the self-documenting expression with the `=` character.

debug.py

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

import math

x = 0.8

print(f'{math.cos(x) = }')
print(f'{math.sin(x) = }')
```

The example outputs the Sine and Cosine functions in the debug mode.

```
$ ./debug.py
math.cos(x) = 0.6967067093471654
math.sin(x) = 0.7173560908995228
```

This is the output.

Ads by Google

Send feedback

Why this ad? ▶

Python multiline f-string

We can work with multiline strings.

multiline.py

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

```
age = 32
occupation = 'gardener'

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

print(msg)
```

The example presents a multiline f-string. The f-strings are placed between round brackets; each of the strings is preceded with the f character.

```
$ python multiline.py
Name: John Doe
Age: 32
Occupation: gardener
```

This is the output.

Python f-string calling function

We can also call functions in f-strings.

call_function.py

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

def mymax(x, y):

    return x if x > y else y

a = 3
b = 4

print(f'Max of {a} and {b} is {mymax(a, b)}')
```

The example calls a custom function in the f-string.

```
$ python call_fun.py
Max of 3 and 4 is 4
```

This is the output.

Python f-string objects

Python f-string accepts objects as well; the objects must have either `__str__()` or `__repr__()` magic functions defined.

objects.py

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

```
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}')
```

The example evaluates an object in the f-string.

```
$ python objects.py
John Doe is a gardener
```

This is the output.

Python f-string escaping characters

The following example shows how to escape certain characters in f-strings.

escaping.py

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

print(f'Python uses {{{}} to evaludate variables in f-strings')
print(f'This was a \'great\' film')
```

To escape a curly bracket, we double the character. A single quote is escaped with a backslash character.

Ads by Google

Send feedback

Why this ad? ▶

```
$ python escaping.py
Python uses {} to evaludate variables in f-strings
This was a 'great' film
```

This is the output.

format_datetime.py

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

import datetime

now = datetime.datetime.now()

print(f'{now:%Y-%m-%d %H:%M}')
```

The example displays a formatted current datetime. The datetime format specifiers follow the : character.

```
$ python format_datetime.py
2019-05-11 22:39
```

This is the output.

Python f-string format floats

Floating point values have the f suffix. We can also specify the precision: the number of decimal places. The precision is a value that goes right after the dot character.

format_floats.py

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

val = 12.3

print(f'{val:.2f}')
print(f'{val:.5f}')
```

The example prints a formatted floating point value.

```
$ python format_floats.py
12.30
12.30000
```

The output shows the number having two and five decimal places.

Python f-string format width

The width specifier sets the width of the value. The value may be filled with spaces or other characters if the value is shorter than the specified width.


format_width.py

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

uses 0 to fill shorter values.

Ads by Google

Send feedback

Why this ad? 

```
$ python format_width.py
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
```

This is the output.

Python f-string justify string

By default, the strings are justified to the left. We can use the > character to justify the strings to the right. The > character follows the colon character.

justify.py

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

```
s1 = 'a'
s2 = 'ab'
s3 = 'abc'
s4 = 'abcd'

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

We have four strings of different length. We set the width of the output to ten characters. The values are justified to the right

This is the output.

Python f-string numeric notations

Numbers can have various numeric notations, such as decadic or hexadecimal.

format_notations.py

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

a = 300

# hexadecimal
print(f"{a:x}")

# octal
print(f"{a:o}")

# scientific
print(f"{a:e}")
```

The example prints a value in three different notations.

```
$ python format_notations.py
12c
454
3.000000e+02
```

This is the output.

In this tutorial, we have worked with Python f-strings.

FREE eBook



5 challenges
to achieving
observability
at scale

[Download now](#)



[Twitter](#) [Github](#) [Subscribe](#) [Privacy](#)

Jan Bodnar admin(at)zetcode.com

Visit [Python tutorial](#) or list [all Python](#) tutorials.