Python

Python f-string
Literal String Interpolation
PEP 498

f-strings in Python

PEP 498 introduced a new string formatting mechanism known as *Literal String Interpolation* or more commonly as *F-strings* (because of the leading *f* character preceding the string literal). The idea behind f-strings is to make string interpolation simpler.

To create an f-string, prefix the string with the letter " f ". The string itself can be formatted in much the same way that you would with str.format(). F-strings provide a concise and convenient way to embed python expressions inside string literals for formatting.

```
# Python3 program introducing f-string
val = 'Geeks'
print(f"{val}for{val} is a portal for {val}.")

name = 'Tushar'
age = 23
print(f"Hello, My name is {name} and I'm {age} years old.")
```

Output :

```
GeeksforGeeks is a portal for Geeks.
Hello, My name is Tushar and I'm 23 years old.
```

Note : F-strings are faster than the two most commonly used string formatting mechanisms, which are % formatting and str.format().

Let’s see few error examples, which might occur while using f-string :

Code #3 : Demonstrating Syntax error.

Python3

```
answer = 456
f"Your answer is "{answer}""
```

Let’s take a look at a basic example:

```
fstring = f'2 + 3 is equal to {2+3}'
```

This will return:

```
2 + 3 is equal to 5.
```

```
height = 2
base = 3
fstring = f'The area of the triangle is
{base*height/2}.'

print(fstring)
```

This returns:

```
The area of the triangle is 3.
```

Accessing Dictionary Items with f-strings

Being able to print out dictionary values within strings makes f-strings even more powerful.

One important thing to note is that you need to be careful to not end your string by using the same type of quote. Let’s take a look at an example:

```
person1 = {
    'name': 'Nik',
    'age': 32,
    'gender': 'male'
}

person2 = {
    'name': 'Katie',
    'age': 30,
    'gender': 'female'
}

fstring = f'{person1.get("name")} is
{person1.get("age")} and is
{person1.get("gender")}.'
print(fstring)
```

This returns:

```
Nik is 32 and is male.
```

```
person1 = {
    'name': 'Nik',
    'age': 32,
    'gender': 'male'
}

person2 = {
    'name': 'Katie',
    'age': 30,
    'gender': 'female'
}

people = [person1, person2]

for person in people:
    print(f'{person.get("name")} is
{person.get("age")} and is
{person.get("gender")}.')
```

This returns:

```
Nik is 32 and is male.
Katie is 30 and is female.
```

Conditionals in Python f-strings

Python f-strings also allow you to evaluate conditional expressions, meaning the result returned is based on a condition.

Let's take a look at a quick, simple example:

```
name = 'Mary'
gender = 'female'

fstring = f'Her name is {name} and {"she"
if gender == "female" else "he"} went to
the store.'
```

This returns:

```
Her name is Mary and she went to the
store.
```

Specifying Alignment with f-strings

To specify alignment with f-strings, you can use a number of different symbols. In order to format strings with alignment, you use any of <, >, ^ for alignment, followed by a digit of values of space reserved for the string. In particular:

- < Left aligned,
- > Right aligned,
- ^ Center aligned.

Let’s take a look at an example:

```
print(f'{"apple" : >30}')
print(f'{"apple" : <30}')
print(f'{"apple" : ^30}')
```

This returns:

```
                         apple
apple
            apple
```