قالب بندی رشته(String Fromat)

رشته‌ها را می‌توانید با استفاده از فرمت‌بندی رشته، اقدام به قرار دادن موارد درون رشته کنید به جای اینکه سعی کنید با استفاده از کاماها یا اتصال رشته‌ها، آیتم‌ها را به هم پیوند دهید. برای مقایسه سریع، در نظر بگیرید:

```
player = 'Thomas'
points = 33

'Last night, '+player+' scored '+str(points)+' points.'  # اتصال
f'Last night, {player} scored {points} points.'          # فرمت‌بندی رشته
```

سه روش برای انجام فرمت‌بندی رشته وجود دارد.

- قدیمی‌ترین روش شامل استفاده از جایگزین‌ها با استفاده از علامت مدولو % است.
- یک تکنیک بهبود یافته از روش .format() رشته استفاده می‌کند.
- جدیدترین روش که با پایتون ۳.۶ معرفی شده است، از رشته‌های قالب‌بندی شده با استفاده از *f-strings* استفاده می‌کند.

از آنجا که احتمالاً با سه نسخه در کد شخص دیگری روبه‌رو می‌شوید، هر کدام را در اینجا توضیح می‌دهیم.

(Placeholders)قالب بندی با استفاده از جا نگه دارها

می‌توانید از %s برای درج رشته‌ها در دستورات چاپ خود استفاده کنید. علامت مدولو % به عنوان "عملگر فرمت‌بندی رشته" شناخته می‌شود.

In [1]:

```
print("I'm going to inject %s here." %'something')
```

```
I'm going to inject something here.
```

می‌توانید چندین آیتم را با قرار دادن آن‌ها درون یک تاپل پس از عملگر % ارسال کنید

In [2]:

```
print("I'm going to inject %s text here, and %s text here." %('some','more'))
```

```
I'm going to inject some text here, and more text here.
```

همچنین می توانید متغییرها را نیز استفاده نمایید.

In [1]:

```
x, y = 'some', 'more'
print("I'm going to inject %s text here, and %s text here."%(x,y))
```

```
I'm going to inject some text here, and more text here.
```

روش های تبدیل قالب

باید توجه شود که دو روش %s و %r با استفاده از دو متد جداگانه str() و repr() هر شیء پایتونی را به یک رشته تبدیل می‌کنند. ما در دوره‌ی بعدی بیشتر درباره این توابع یاد خواهیم گرفت، اما باید توجه داشت که %r و repr() نمایش رشته مربوط به شیء را ارائه می‌دهند که شامل نقل قول و هر کاراکتر خروجی است.

In [4]:

```
print('He said his name was %s.' %'Fred')
print('He said his name was %r.' %'Fred')
```

```
He said his name was Fred.
He said his name was 'Fred'.
```

به عنوان مثال دیگر، \t یک تب در یک رشته وارد می‌کند.

In [5]:

```
print('I once caught a fish %s.' %'this \tbig')
print('I once caught a fish %r.' %'this \tbig')
```

```
I once caught a fish this	big.
I once caught a fish 'this \tbig'.
```

عامل %s هر آنچه را که می‌بیند به رشته تبدیل می‌کند، از جمله اعداد صحیح و اعشاری. عامل %d ابتدا اعداد را به صورت اعداد صحیح تبدیل می‌کند، بدون گرد کردن. تفاوت را در زیر مشاهده کنید.

In [6]:

```
print('I wrote %s programs today.' %3.75)
print('I wrote %d programs today.' %3.75)
```

```
I wrote 3.75 programs today.
I wrote 3 programs today.
```

فاصله گذاری و مشخص کردن دقت در اعداد اعشاری

اعداد اعشاری از فرمت %5.2f استفاده می‌کنند. در اینجا، عدد 5 حداقل تعداد کاراکترهایی است که رشته باید شامل شود؛ این اعداد می‌توانند با فضای سفید پر شوند اگر کل عدد به اندازه این تعداد رقم نداشته باشد. در کنار این، .2f برای نشان دادن چند رقم اعشاری پس از نقطه اعشاری استفاده می‌شود. بیایید چند مثال ببینیم:

In [7]:

```
print('Floating point numbers: %5.2f' %(13.144))
```

```
Floating point numbers: 13.14
```

In [8]:

```
print('Floating point numbers: %1.0f' %(13.144))
```

```
Floating point numbers: 13
```

In [9]:

```
print('Floating point numbers: %1.5f' %(13.144))
```

```
Floating point numbers: 13.14400
```

In [10]:

```
print('Floating point numbers: %10.2f' %(13.144))
```

```
Floating point numbers:      13.14
```

In [11]:

```
print('Floating point numbers: %25.2f' %(13.144))
```

```
Floating point numbers:                     13.14
```

برای کسب اطلاعات بیشتر در مورد قالب‌بندی رشته با استفاده از جایگزین‌ها، به لینک زیر مراجعه کنید:

https://docs.python.org/3/library/stdtypes.html#old-string-formatting
(https://docs.python.org/3/library/stdtypes.html#old-string-formatting)

In [12]:

```
print('First: %s, Second: %5.2f, Third: %r' %('hi!',3.1415,'bye!'))
```

```
First: hi!, Second:  3.14, Third: 'bye!'
```

قالب بندی رشته ها با متد .format()

یک روش بهتر برای قالب‌بندی اشیاء در رشته‌هایتان برای دستورهای چاپ، استفاده از روش .format() است. سینتکس آن به شرح زیر است:

```
'String here {} then also {}'.format('something1','something2')
```

به عنوان مثال:

In [13]:

```
print('This is a string with an {}'.format('insert'))
```

```
This is a string with an insert
```

روش .format() چندین مزیت نسبت به روش placeholder %s دارد:

1. **ترتیب‌بندی**: با استفاده از روش .format() می‌توانید ترتیب جایگشت placeholderها را به صورت صریح با استفاده از اندیس‌ها یا نام‌ها مشخص کنید. این امکان را به شما می‌دهد تا به طور انعطاف‌پذیرتری آرگومان‌ها را درون رشته ترتیب دهید.

مقادیر اشیا ورودی را می توانیم با ایندکس گذاری با ترتیب های متفاوتی در رشته قرار دهیم

In [14]:

```
print('The {2} {1} {0}'.format('fox','brown','quick'))
```

```
The quick brown fox
```

مقادیر اشیا ورودی را می توانیم با کلمات کلیدی خاص استفاده نماییم

In [15]:

```
('First Object: {a}, Second Object: {b}, Third Object: {c}'.format(a=1,b='Two',c=12.3))
```

```
First Object: 1, Second Object: Two, Third Object: 12.3
```

مقادیر اشیا ورودی را می توانیم چندین بار مورد استفاده قرار دهیم

In [16]:

```
print('A %s saved is a %s earned.' %('penny','penny'))
# vs.
print('A {p} saved is a {p} earned.'.format(p='penny'))
```

```
A penny saved is a penny earned.
A penny saved is a penny earned.
```

جهت گذاری، فاصله گذاری و نمایش دقت با متد `.format()`

در داخل آکولاد های می‌توانید طول فیلد، ترازبندی چپ/راست، پارامترهای گردکردن و سایر موارد را تعیین کنید.

In [17]:

```
print('{0:8} | {1:9}'.format('Fruit', 'Quantity'))
print('{0:8} | {1:9}'.format('Apples', 3.))
print('{0:8} | {1:9}'.format('Oranges', 10))
```

```
Fruit    | Quantity
Apples   |       3.0
Oranges  |        10
```

به طور پیش‌فرض، `.format()` متن را به چپ و اعداد را به راست تراز می‌کند. می‌توانید از `<` ، `>` یا `^` به عنوان اختیاری استفاده کنید تا ترازبندی چپ، مرکز یا راست را تعیین کنید.

In [18]:

```
print('{0:<8} | {1:^8} | {2:>8}'.format('Left','Center','Right'))
print('{0:<8} | {1:^8} | {2:>8}'.format(11,22,33))
```

```
Left     |  Center  |    Right
11       |    22    |       33
```

می‌توانید قبل از عملگر ترازبندی یک کاراکتر پر کننده قرار دهید.

In [19]:

```
print('{0:=<8} | {1:-^8} | {2:.>8}'.format('Left','Center','Right'))
print('{0:=<8} | {1:-^8} | {2:.>8}'.format(11,22,33))
```

```
Left==== | -Center- | ...Right
11====== | ---22--- | ......33
```

عرض فیلدها و دقت اعداد اعشاری به یک روش مشابه با placeholder ها مدیریت می‌شوند. دو دستور چاپ زیر معادل هم هستند:

In [20]:

```
print('This is my ten-character, two-decimal number:%10.2f' %13.579)
print('This is my ten-character, two-decimal number:{0:10.2f}'.format(13.579))
```

```
This is my ten-character, two-decimal number:     13.58
This is my ten-character, two-decimal number:     13.58
```

توجه کنید که پس از علامت دونقطه ۵ فاصله وجود دارد و عدد ۱۳.۵۸ پنج کاراکتر را به خود اختصاص داده است، بنابراین مجموعاً ده کاراکتر تشکیل می‌دهد.

برای کسب اطلاعات بیشتر در مورد روش `.format()` رشته، به لینک زیر مراجعه کنید:

https://docs.python.org/3/library/string.html#formatstrings
(https://docs.python.org/3/library/string.html#formatstrings)

Literals (f-strings)قالب بندی رشته ها با

در Python 3.6 معرفی شده‌اند، f-strings چندین مزیت نسبت به روش قدیمی `.format()` رشته که در بالا توضیح داده شده است، دارند. یکی از مزایای آنها این است که می‌توانید متغیرهای بیرونی را به طور مستقیم درون رشته قرار دهید، بدون اینکه آنها را به عنوان آرگومان‌ها از طریق `.format(var)` ارسال کنید.

In [2]:

```
name = 'Fred'

print(f"He said his name is {name}.")
```

```
He said his name is Fred.
```

برای دریافت نمایش رشته‌ای یک شیء، می‌توانید `!r` را درون f-string استفاده کنید:

In [22]:

```
print(f"He said his name is {name!r}")
```

```
He said his name is 'Fred'
```

`"result: {value:{width}.{precision}}"` قالب بندی اعداد اعشاری

در روش `.format()` شما ممکن است بنویسید `{value:10.4f}` ، اما با استفاده از f-strings این می‌تواند به صورت `{value:{10}.{6}}` تبدیل شود.

In [23]:

```
num = 23.45678
print("My 10 character, four decimal number is:{0:10.4f}".format(num))
print(f"My 10 character, four decimal number is:{num:{10}.{6}}")
```

```
My 10 character, four decimal number is:   23.4568
My 10 character, four decimal number is:   23.4568
```

توجه داشته باشید که با استفاده از f-strings، دقت به تعداد کل ارقام اعشاری اشاره می‌کند، نه فقط آنچه که پس از ممیز قرار می‌گیرد. این مطابقت بیشتری با نمایش علمی و تجزیه و تحلیل آماری دارد. متأسفانه، در f-strings، حتی اگر دقت این امکان را فراهم کند، پدینگ به سمت راست اعشاری انجام نمی‌شود.

In [24]:

```
num = 23.45
print("My 10 character, four decimal number is:{0:10.4f}".format(num))
print(f"My 10 character, four decimal number is:{num:{10}.{6}}")
```

```
My 10 character, four decimal number is:   23.4500
My 10 character, four decimal number is:     23.45
```

اگر این موضوع مهم هست، همیشه می‌توانید از سینتکس روش `.format()` درون یک f-string استفاده کنید.

In [25]:

```
num = 23.45
print("My 10 character, four decimal number is:{0:10.4f}".format(num))
print(f"My 10 character, four decimal number is:{num:10.4f}")
```

```
My 10 character, four decimal number is:   23.4500
My 10 character, four decimal number is:   23.4500
```

برای مطالب بیشتر به لینک مقابل مراجعه فرمایید.https://docs.python.org/3/reference/lexical_analysis.html#f-strings (https://docs.python.org/3/reference/lexical_analysis.html#f-strings)

Template Strings

In [3]:

```
from string import Template
```

In [4]:

```
t = Template('Hey, $name!')
```

In [5]:

```
s = 'John'

t.substitute(name = s)
```

Out[5]:

```
'Hey, John!'
```

اینم از بحث قالب بندی رشته ها!