String (رشته)

سلام! در این بخش، قصد داریم با رشته‌ها (Strings) در پایتون آشنا شویم. رشته‌ها مجموعه‌ای از کاراکترها هستند که داخل علامت نقل قول (' یا ") قرار می‌گیرند. برای مثال:

Copy Plain text

```
text = "سلام، به کارگاه برنامه‌سازی پیشرفته خوش آمدید!"
```

می‌توان گفت رشته‌ها از محبوب‌ترین نوع داده‌ها در پایتون هستند. با استفاده از رشته‌ها می‌توان کارهای زیادی را با دستورهای ساده انجام داد.

تعریف کردن String

همانطور که اشاره کردیم، برای تعریف یک رشته در پایتون، کافی است آن را داخل " یا ' قرار دهید. حتی خروجی دستور `input()` که برای گرفتن ورودی از کاربر استفاده می‌شود، یک رشته است. می‌توانید رشته‌ها را به متغیرها اختصاص دهید. به این مثال‌ها توجه کنید:

```
s = "Hello "
t = '!'
name = input()
print(s + name + t)
```

نکته نکته جالب این است که رشته‌ها در پایتون immutable هستند، یعنی بعد از ساختن‌شان نمی‌توانیم مقدارشان را تغییر دهیم. برای تغییر یک رشته، باید یک رشته جدید بسازیم و مقدار رشته قبلی را به رشته جدید اختصاص دهیم. مثلاً در زبان C می‌توانید هر کاراکتر از یک رشته را تغییر دهید، ولی در پایتون این امکان وجود ندارد. مثال:

```
s = "Arman"
s[2] = 'i' #TypeError: 'str' object does not support item assignment
print(s)
```

عملگرهای String

در اینجا به تعدادی از عملگرهای کاربردی‌تر رشته اشاره می‌کنیم که در صورت کار با رشته معمولاً به آن‌ها نیاز خواهیم داشت.

عملگر	عملکرد
`+`	دو رشته را به هم می‌چسباند (concat) و نتیجه را به عنوان خروجی برمی‌گرداند.
`*`	یک رشته را به تعداد مشخصی پشت سر هم می‌گذارد و برمی‌گرداند
`in`	اگر رشته قبل از آن در رشته بعد از آن موجود باشد `True` و در غیر این صورت `False` بر می‌گرداند
`not in`	دقیقا برعکس عملگر `in` کار می‌کند

برای درک این موضوع به این مثال‌ها توجه کنید:

```
>>> s = 'salam'
>>> t = 'ali'
>>> s + t
'salamali'
>>> s * 3
'salamsalamsalam'
>>> 'ai' in s
False
>>> 'al' in s
True
>>> 'ai' not in s
True
>>> 'al' not in s
False
```

روش‌های دسترسی به خانه‌های String

۱. دسترسی به یک خانه

با استفاده از `s[i]` می‌توانید به خانه iام رشته `s` دسترسی پیدا کنید و مقدار اون رو بخونید. . یادتان باشد که در پایتون، شماره‌گذاری اندیس‌ها همانند دیگر زبان‌ها از صفر شروع می‌شود، یعنی اولین حرف رشته `s` برابر با `s[0]` است. در این مثال ببینید:

```
>>> s = 'mokar'
>>> s[0]
m
>>> s[2]
k
>>> s[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> s[0] = 'b'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

توجه

- عضو iام در عملگر `[i]` باید در رشته قبل از خودش موجود باشه، وگرنه IndexError میده.
- به این علت که رشته immutable است نمی‌تونیم عضو iام را تغییر دهیم.

۲. دسترسی به چند خانه

برای این کار می‌تونید از عملگر `[:]` استفاده کنید. این عملگر دو عدد صحیح `l` و `r` رو به عنوان ورودی می‌گیره و رشته‌ای که از عضو `l` ام تا `r` ام هست رو برمی‌گردونه. توجه کنید که اگر `l` رو وارد نکنید، به طور پیش‌فرض از اندیس ۰ شروع میشه و اگر `r` رو وارد نکنید، تا انتهای رشته رو می‌گیره. همچنین اگر یکی از این اندیس‌ها منفی باشه، به جای اون مقدار، **طول رشته منهای اندیس** قرار می‌گیره. ببینید:

```
>>> s = 'salam'
>>> s[1:4]
```

```
'ala'
>>> s[1:]
'alam'
>>> s[:3]
'sal'
>>> s[4:3]
''
>>> s[:]
'salam'
>>> s[-3:-1]
'la'
>>> s[-3:]
'lam'
>>> s[:-1]
'sala'
>>> s[-3:4]
'la'
>>> s[-4:3]
'al'
```

توابع String

این توابع به طور مستقیم روی رشته‌ها اعمال می‌شوند و نتیجه رو به شما برمی‌گردونند. در اینجا برخی از توابع پرکاربرد رشته‌ها رو معرفی می‌کنیم:

عملگر	عملکرد
`lower()`	تمام حروف رشته را به حروف کوچک تبدیل کرده و بر می‌گرداند
`upper()`	تمام حروف رشته را به حروف بزرگ تبدیل کرده و بر می‌گرداند
`isalpha()`	در صورتی که تمام کاراکترهای رشته حروف الفبا باشند `True` و در غیر این صورت `False` بر می‌گرداند
`isdigit()`	در صورتی که تمام کاراکترهای رشته رقم باشند `True` و در غیر این صورت `False` بر می‌گرداند
`zfill(x)`	تا زمانی که طول رشته اصلی کمتر از `x` باشد در ابتدای آن `0` قرار می‌دهد

```
>>> 'Ali23'.lower()
'ali23'
>>> 'Ali23'.upper()
'ALI23'
>>> 'Ali23'.isalpha()
False
>>> 'Ali'.isalpha()
True
>>> 'Ali23'.isdigit()
False
>>> '23'.isdigit()
True
>>> 'Ali23'.zfill(2)
'Ali23'
>>> 'Ali23'.zfill(8)
'000Ali23'
```

عملگر	عملکرد
`count()`	تعداد دفعات تکرار رشته ورودی را در رشته اصلی بر می‌گرداند.
`find()`	رشته‌ای را گرفته و مکان اولین تکرار آن را در رشته اصلی بر می‌گرداند. در صورتی‌که تکراری وجود نداشت `-1` را باز می‌گرداند
`split()`	یک رشته گرفته و در مکان‌های تکرار آن (در رشته اصلی)، رشته اصلی را شکسته و زیررشته‌های به دست آمده را در قالب یک لیست بر می‌گرداند(کاربرد این تابع را در دریافت ورودی از کاربر از درسنامه کارگاه ۱ دیدیم.)
`replace()`	دو رشته گرفته و هرجایی که رشته اول در رشته اصلی یافت بشود آن را با رشته دوم جایگزین می‌کند.

```
>>> 'ahmad'.count('ma')
1
>>> 'ahmad'.count('ali')
0
>>> 'aaaa'.count('aa')
2
>>> 'ahmad'.find('ali')
```

```
-1
>>> 'ahmad'.find('ma')
2
>>> 'Hello world!'.split(' ')
['Hello', 'world!']
>>> 'Hello world!'.split('l')
['He', '', 'o wor', 'd!']
>>> 'Hello world!'.replace('l', 'salam!')
'Hesalam!salam!o worsalam!d!'
>>> 'Hello world!'.replace('h', 'salam!')
'Hello world!'
>>> 'HeLLo'.replace('LL', 'll')
Hello
```

تابع `len()`

یکی از توابع مهم در کار با رشته‌ها، تابع len() است که تعداد کاراکترهای یک رشته را برمی‌گرداند:

```
>>> s = "ali"
>>> len(s)
3
>>> len("salam")
5
```

یادآوری تغییرناپذیری String

همانطور که قبلاً گفته شد، رشته‌ها در پایتون تغییرناپذیر هستند، بنابراین تمامی توابعی که در بالا ذکر کردیم روی رشته اعمال نمی‌شوند و تنها نتیجه را برمی‌گردانند. اگر بخواهید رشته‌ای را تغییر دهید، باید یک رشته جدید بسازید. مثلاً:

```
>>> s = 'mokar'
>>> s.replace('m', 'l').upper()
LOKAR
>>> s
mokar
>>> s = s.replace('m', 'l').upper()
```

```
>>> s
LOKAR
```