رشته ها

رشته‌ها در پایتون برای ثبت اطلاعات متنی مانند نام‌ها استفاده می‌شوند. رشته‌ها در پایتون در واقع یک *دنباله* هستند، که به اصطلاح به این معناست که پایتون هر عنصر را در رشته به عنوان یک دنباله در نظر می‌گیرد. به عنوان مثال، پایتون رشته "hello" را به عنوان یک دنباله از حروف با ترتیب خاص درک می‌کند. این بدان معناست که ما قادر خواهیم بود با استفاده از ایندکس‌گذاری، حروف خاصی را دریافت کنیم (مانند اولین حرف یا آخرین حرف).

این ایده از دنباله یک مفهوم مهم در پایتون است و ما در آینده به آن خواهیم پرداخت.

در این درس، درباره موارد زیر آموخته خواهیم شد:

1. ایجاد رشته‌ها
2. چاپ رشته‌ها
3. ایندکس‌گذاری و برش رشته‌ها
4. ویژگی‌های رشته‌ها
5. متدهای رشته‌ها
6. قالب‌بندی چاپ

ایجاد رشته

برای تعریف یا ایجاد رشته ها در پایتون باید متن مورد نظر خود را داخل جفت تک کوتیشن یا دابل کوتیشن قرار بدهید

In [15]:

```
# Single word
'Hello'
```

Out[15]:

```
'Hello'
```

In [16]:

```
type('Hello')
```

Out[16]:

```
str
```

In [17]:

```
# Entire phrase
'This is also a string!'
```

Out[17]:

```
'This is also a string!'
```

In [18]:

```
# We can also use double quote
"String built with double qoutes"
```

Out[18]:

```
'String built with double qoutes'
```

In [19]:

```
# Be careful with quotes!
' I'm using single quotes, but this will create an error'
```

```
  Cell In[19], line 2
    ' I'm using single quotes, but this will create an error'
                                                             ^
SyntaxError: unterminated string literal (detected at line 2)
```

دلیل بروز خطا در بالا به این دلیل است که نقطه تکی کوتیشن در `I'm` رشته را متوقف کرد. شما می‌توانید ترکیبی از نقطه دوتایی و نقطه تکی را برای به دست آوردن عبارت کامل استفاده کنید.

In [20]:

```
"I'm using single quotes, but this will create an error"
```

Out[20]:

```
"I'm using single quotes, but this will create an error"
```

In [21]:

```
' I\'m using single quotes, but this will create an error'
```

Out[21]:

```
" I'm using single quotes, but this will create an error"
```

In [23]:

```
print("I\tm using single quotes, but this will create an error")
```

```
I       m using single quotes, but this will create an error
```

حالا در ادامه درباره نحوه چاپ کردن (print) یاد بگیریم

چاپ یا print کردن رشته ها

با استفاده از دفترچه Jupyter با فقط یک رشته در یک سلول، به طور خودکار رشته‌ها را در خروجی نمایش می‌دهد، اما روش صحیح برای نمایش رشته‌ها در خروجی شما استفاده از تابع چاپ (print) است.

In [24]:

```
# We can simply declare a string
'Hello world!'
```

Out[24]:

```
'Hello world!'
```

In [25]:

```
# Note that we can't output multiple strings this way
'Hello world! 1'
'Hello world! 2'
```

Out[25]:

```
'Hello world! 2'
```

ما از دستور print می توان چاپ چندین رشته استفاده نماییم.

In [28]:

```
print('Hello world! 1')
print('Hello world! 2')

print('Use \n to print a new line')
print('\n')
print('See what i mean?')
```

```
Hello world! 1
Hello world! 2
Use
 to print a new line


See what i mean?
```

In [30]:

```
print('Hello', 'World', 555)
```

```
Hello World 555
```

In [32]:

```
a = 5

a = a ** \
2
```

مفاهیم پایه رشته ها

استفاده از تابع len() برای بدست آوردن تعداد کاراکترهای (طول) رشته

In [33]:

```
len('Hello world!')
```

Out[33]:

```
12
```

تابع len() یکی از توابع داخلی پایتون می باشد که تمامی کاراکترها را در رشته شمارش می‌کند، شامل فاصله‌ها و علائم نگارشی.

ایندکس گذاری رشته ها

ما می‌دانیم رشته‌ها یک دنباله هستند، به این معنا که پایتون می‌تواند از ایندکس‌ها برای فراخوانی بخش‌های مختلف دنباله استفاده کند. بیایید ببینیم این عملیات چگونه کار می‌کند.

در پایتون، ما از براکت‌ها [ ] پس از یک شیء استفاده می‌کنیم تا ایندکس آن را فراخوانی کنیم. همچنین باید توجه کنیم که در پایتون، شماره‌گذاری از 0 شروع می‌شود. بیایید یک شیء جدید به نام s ایجاد کنیم و سپس از چند مثال از ایندکس‌گذاری استفاده کنیم.

In [34]:

```
# Assign s as a string
s = "Hello world!"
```

In [35]:

```
#Check
s
```

Out[35]:

```
'Hello world!'
```

In [36]:

```
# Print the object
print(s)
```

```
Hello world!
```

In [37]:

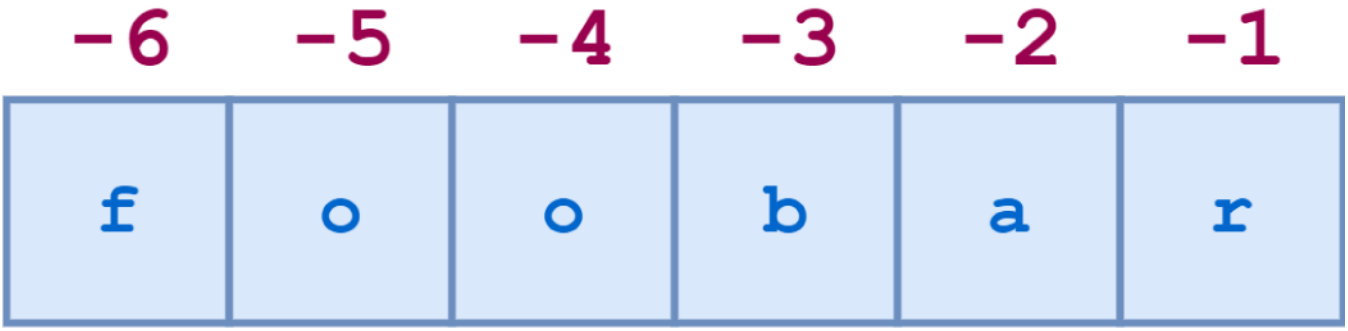
```
s = 'foobar'
```

In [ ]:

```

```

حالا بیایید از ایندکس گذاری در رشته ها استفاده کنیم!



In [38]:

```
# Show first element (in this case a letter)
s[0]
```

Out[38]:

```
'f'
```

In [39]:

```
s[2]
```

Out[39]:

```
'o'
```

In [40]:

```
s[5]
```

Out[40]:

```
'r'
```

In [41]:

```
s[10]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[41], line 1
----> 1 s[10]

IndexError: string index out of range
```

ما می‌توانیم از علامت : برای انجام برش (slicing) استفاده کنیم که همه چیز را تا یک نقطه مشخص انتخاب کنیم. به عنوان مثال:

In [42]:

```
# Grab everything past the first term all the way to the length of s which is len(s)
#s[start:stop:step]
s[0:3]
```

Out[42]:

```
'foo'
```

In [43]:

```
# Note that there is no change to the original s
s[2:5]
```

Out[43]:

```
'oba'
```

In [44]:

```
# Grab everything UP TO the 3rd index
s[2:]
```

Out[44]:

```
'obar'
```

توجه کنید که در برش بالا، ما به پایتون می‌گوییم همه چیز را از 0 تا 3 بگیرد. اما این شامل ایندکس 3 نمی‌شود. شما این را در پایتون بسیار مشاهده خواهید کرد، که عبارت‌ها و معمولاً در زمینه "تا، اما شامل نشده" هستند.

In [45]:

```
#Everything
s[:]
```

Out[45]:

```
'foobar'
```

همچنین ما می توانیم از مقادیر منفی برای ایندکس گذاری استفاده کنیم که در این حالت مقادیر از انتهای رشته انتخاب خواهند شد

In [46]:

```
# Last letter (one index behind 0 so it loops back around)
s[-1]
```

Out[46]:

```
'r'
```

In [47]:

```
s[-2]
```

Out[47]:

```
'a'
```

In [48]:

```
# Grab everything but the last letter
s[:-1]
```

Out[48]:

```
'fooba'
```

ما همچنین می‌توانیم از نماد ایندکس و برش برای گرفتن عناصر یک دنباله با اندازه گام مشخص استفاده کنیم (پیش‌فرض 1 است). به عنوان مثال، ما می‌توانیم از دو : و سپس یک عدد مشخص کننده فرکانس را برای گرفتن عناصر استفاده کنیم. به عنوان مثال:

In [49]:

```
# Grab everything, but go in steps size of 1
s[::1]
```

Out[49]:

```
'foobar'
```

In [50]:

```
# Grab everything, but go in step sizes of 2
s[::2]
```

Out[50]:

```
'foa'
```

In [51]:

```
# We can use this to print a string backwards
s[::-1]
```

Out[51]:

```
'raboof'
```

ویژگی های رشته ها

مهم است که توجه کنیم رشته‌ها دارای ویژگی مهمی به نام *غیرقابل تغییری* (immutability) هستند. این بدان معناست که یکبار رشته ایجاد شود، عناصر درون آن قابل تغییر یا جایگزینی نیستند. به عنوان مثال:

In [52]:

```
s[0]
```

Out[52]:

```
'f'
```

In [53]:

```
# Let's try to change the first letter to 'x'
s[0] = 'X'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[53], line 2
      1 # Let's try to change the first letter to 'x'
----> 2 s[0] = 'X'

TypeError: 'str' object does not support item assignment
```

In [54]:

```
s = 'Hello'
```

In [55]:

```
s
```

Out[55]:

```
'Hello'
```

توجه کنید که خطای به ما می‌گوید که چه کاری نمی‌توانیم انجام دهیم، تغییر دادن عنصر!

اما چیزی که می‌توانیم انجام دهیم، اتصال (concatenate) رشته‌هاست!

In [56]:

```
s = "hello world!"
```

In [57]:

```
# Concatenate strings!
s + " Concatenate Me!!"
```

Out[57]:

```
'hello world! Concatenate Me!!'
```

In [58]:

```
s + 5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[58], line 1
----> 1 s + 5

TypeError: can only concatenate str (not "int") to str
```

In [59]:

```
# We can reassign s completely though!
s
```

Out[59]:

```
'hello world!'
```

In [60]:

```
s = s + " Concatenate Me!!"
```

In [61]:

```
print(s)
```

```
hello world! Concatenate Me!!
```

همچنین از علامت ضرب نیز می توانیم برای تکرار یک رشته استفاده نماییم.

In [62]:

```
letter = 'z'
```

In [66]:

```
letter * 100
```

Out[66]:

```
'zzzz
```

که در آن پارامترها آرگومان‌های اضافی‌ای هستند که می‌توانیم به متد ارسال کنیم. نگران نباشید اگر جزئیات در حال حاضر ۱۰۰٪ متوجه نشوید. در آینده، خودمان شی و تابع‌های خود را خواهیم ساخت!

در زیر چند نمونه از متدهای داخلی رشته‌ها آورده شده است:

In [67]:

```
s = "Hello World!"
s[0]
```

Out[67]:

```
'H'
```

In [69]:

```
"Hello World!"[0::2]
```

Out[69]:

```
'HloWrd'
```

In [72]:

```
# Upper Case a string
x = s.upper()
x
```

Out[72]:

```
'HELLO WORLD!'
```

In [71]:

```
s
```

Out[71]:

```
'Hello World!'
```

In [73]:

```
# Lower case
s.lower()
```

Out[73]:

```
'hello world!'
```

In [74]:

```
s.capitalize()
```

Out[74]:

```
'Hello world!'
```

In [76]:

```
s
```

Out[76]:

```
'Hello World!'
```

In [80]:

```
s.count('ll')
```

Out[80]:

```
1
```

In [81]:

```
s.replace('l', '3')
```

Out[81]:

```
'He33o Wor3d!'
```

In [82]:

```
s
```

Out[82]:

```
'Hello World!'
```

In [83]:

```
s1 = '    string number one      '
```

In [84]:

```
s1.strip()
```

Out[84]:

```
'string number one'
```

In [85]:

```
s1.lstrip()
```

Out[85]:

```
'string number one      '
```

In [86]:

```
s1.rstrip()
```

Out[86]:

```
'    string number one'
```

In [88]:

```
s
```

Out[88]:

```
'Hello World!'
```

In [87]:

```
# Split a string by blank space (this is the default)
s.split()
```

Out[87]:

```
['Hello', 'World!']
```

In [90]:

```
# Split by a specific element (doesn't include the element that was split on)
s.split('ll')
```

Out[90]:

```
['He', 'o World!']
```

مرحله بعدی: قالب بندی رشته ها(string fromatting)