

تاپل‌ها

در پایتون، تاپل‌ها بسیار شبیه به لیست‌ها هستند، با این تفاوت که، برخلاف لیست‌ها، آنها غیرقابل تغییر هستند و یعنی نمی‌توان آنها را تغییر داد. شما از تاپل‌ها برای نمایش اشیاءى که نباید تغییر کنند، مانند روزهای هفته یا تاریخ‌ها در تقویم استفاده خواهید کرد.

در این بخش، مروری کوتاهی بر موارد زیر خواهیم داشت:

1. ساخت تاپل‌ها
2. متدهای پایه‌ای تاپل
3. غیرقابل تغییر بودن
4. زمان استفاده از تاپل‌ها

بر اساس آنچه درباره لیست‌ها یاد گرفته‌اید، شما به نحوی به تاپل‌ها آشنا خواهید بود. ما می‌توانیم با رفتار بسیار مشابهی از آنها استفاده کنیم، با این تفاوت که تاپل‌ها غیرقابل تغییر هستند.

ساخت تاپل‌ها

ساخت تاپل‌ها با استفاده از پرانتز () و عناصری که توسط کاما از هم جدا شده‌اند، انجام می‌شود. به عنوان مثال:

In [5]:

```
1 # Create a tuple
2 t = (1, 2, 3, 4, 5)
```

In [6]:

```
1 type(t)
```

Out[6]:

tuple

In [7]:

```
1 # Check len just like a list
2 len(t)
```

Out[7]:

5

In [22]:

```
1 # Can also mix object types
2 t = ('one', 2, 'one')
3
4 # Show
5 t
```

Out[22]:

('one', 2, 'one')

In [9]:

```
1 # Use indexing just like we did in lists
2 t[0]
```

Out[9]:

'one'

In [10]:

```
1 # Slicing just like a list
2 t[-1]
```

Out[10]:

2

تعریف تاپل تک عضوی

In [11]:

```
1 x = (1)
```

In [12]:

```
1 x
```

Out[12]:

1

In [13]:

```
1 type(x)
```

Out[13]:

int

In [17]:

```
1 x = (1,)
```

In [18]:

```
1 type(x)
```

Out[18]:

tuple

In [19]:

```
1 x
```

Out[19]:

```
(1,)
```

متدهای پایه تاپل ها

تاپل ها دارای متدهای داخلی هستند، اما تعداد آنها به اندازه لیست ها نیست. بیایید به دو متد آنها نگاهی بیندازیم:

In [21]:

```
1 # Use .index to enter a value and return the index
2 t.index(2)
```

Out[21]:

```
1
```

In [23]:

```
1 t
```

Out[23]:

```
('one', 2, 'one')
```

In [24]:

```
1 # Use .count to count the number of times a value appears
2 t.count('one')
```

Out[24]:

```
2
```

Immutabilityغیرقابل جهش بودن

اهمیت غیرقابل تغییر بودن تاپل ها را نمی توان به اندازه کافی تاکید کرد. برای بیان این نکته:

In [25]:

```
1 t[0] = 'change'
```

TypeError

Traceback (most recent call last)

t)

Cell In[25], line 1

----> 1 t[0] = 'change'

TypeError: 'tuple' object does not support item assignment

به دلیل این غیرقابل تغییری، تاپل‌ها قادر به رشد نیستند. یکباری که یک تاپل ساخته شد، نمی‌توانیم به آن عناصر اضافه کنیم.

In [26]:

```
1 t.append('nope')
```

```
-----  
-  
AttributeError                                Traceback (most recent call las  
t)  
Cell In[26], line 1  
----> 1 t.append('nope')
```

AttributeError: 'tuple' object has no attribute 'append'

نکته بسیار مهم، تاپل‌ها فقط در سطح اول خود غیرقابل جهش می‌باشند و عملکرد سطوح بعدی به نوع داده آن سطح بستگی دارد

In [28]:

```
1 t = (1, 2, 3, (5,6), [10, 11, 12])
```

In [30]:

```
1 t[0] = 55
```

```
-----  
-  
TypeError                                Traceback (most recent call las  
t)  
Cell In[30], line 1  
----> 1 t[0] = 55
```

TypeError: 'tuple' object does not support item assignment

In [31]:

```
1 t[3]
```

Out[31]:

(5, 6)

In [33]:

```
1 t[3][1] = 55
```

TypeError

Traceback (most recent call last)

t)

Cell In[33], line 1

```
----> 1 t[3][1] = 55
```

TypeError: 'tuple' object does not support item assignment

In [34]:

```
1 t[3] = (555, 666)
```

TypeError

Traceback (most recent call last)

t)

Cell In[34], line 1

```
----> 1 t[3] = (555, 666)
```

TypeError: 'tuple' object does not support item assignment

In [35]:

```
1 t
```

Out[35]:

```
(1, 2, 3, (5, 6), [10, 11, 12])
```

In [36]:

```
1 t[4]
```

Out[36]:

```
[10, 11, 12]
```

In [38]:

```
1 t[4] = [55, 66]
```

TypeError

Traceback (most recent call last)

t)

Cell In[38], line 1

```
----> 1 t[4] = [55, 66]
```

TypeError: 'tuple' object does not support item assignment

In [40]:

```
1 t[4][0] = -100
```

In [41]:

```
1 t
```

Out[41]:

```
(1, 2, 3, (5, 6), [-100, 11, 12])
```

بسته بندی و بازکردن بسته در تاپل ها

In [42]:

```
1 # tuple packing
2 x = 1, 'one', 2, 'three'
```

In [43]:

```
1 x
```

Out[43]:

```
(1, 'one', 2, 'three')
```

In [44]:

```
1 type(x)
```

Out[44]:

```
tuple
```

In [45]:

```
1 # tuple unpacking
2 x1, x2, x3, x4 = x
3
4 print(x1, x2, x3, x4)
```

```
1 one 2 three
```

In [46]:

```
1 # be careful
2
3 x1, x2, x3 = x
```

```
-----
-
ValueError                                Traceback (most recent call las
t)
Cell In[46], line 3
      1 # be careful
----> 3 x1, x2, x3 = x

ValueError: too many values to unpack (expected 3)
```

In [47]:

```
1 x1, x2, x3, x4, x5 = x
```

```
-----
-
ValueError                                Traceback (most recent call las
t)
Cell In[47], line 1
----> 1 x1, x2, x3, x4, x5 = x

ValueError: not enough values to unpack (expected 5, got 4)
```

چه زمانی از تاپل ها

شاید بپرسید: "چرا باید از تاپل ها استفاده کنیم وقتی که تعداد کمتری از متدها در دسترس هستند؟" صادقانه بگوییم، تاپل ها به همان اندازه که لیست ها در برنامه نویسی استفاده می شوند، اما زمانی که غیرقابل تغییر بودن ضروری است، تاپل ها مورد استفاده قرار می گیرند. اگر در برنامه خود شما یک شیء را انتقال می دهید و نیاز دارید مطمئن شوید که آن تغییر نمی کند، در این صورت تاپل بهترین گزینه است. آنها منبع مناسبی از سلامت داده را فراهم می کنند.

اکنون باید قادر باشید تاپل ها را ایجاد و استفاده کنید و درکی از غیرقابل تغییری آنها داشته باشید.

در ادامه، دیکشنری ها!