دستورات تو در تو و محدوده (Nested Statements and Scope)

حالا که ما در مورد نوشتن توابع خودمان صحبت کردیم، مهم است که بدانیم پایتون چگونه با نام‌های متغیری که شما تعیین می‌کنید برخورد می‌کند. زمانی که شما یک نام متغیر در پایتون ایجاد می‌کنید، این نام در یک *فضای نام* ذخیره می‌شود.
نام‌های متغیر همچنین یک محدوده یا *scope* دارند، این محدوده قابلیت دیدن آن نام متغیر را برای بخش‌های دیگر کد شما تعیین می‌کند.

بیایید با یک تجربه فکری سریع شروع کنیم؛ تصور کنید کد زیر را:

In [2]:

```
x = 25

def printer():
    x = 50
    return x

printer()
# print(x)
# print(printer())
```

Out[2]:

```
50
```

بنظر شما خروجی تابع `printer()` چه خواهد بود؟ 25 یا 50؟ خروجی چاپ x در سطر 7 چطور؟ 25 یا 50؟

In [3]:

```
print(x)
```

```
25
```

In [4]:

```
print(printer())
```

```
50
```

جالب است! اما پایتون چگونه می‌داند که شما در کد خود به کدام **x** اشاره می‌کنید؟ در اینجا ایده محدوده وارد می‌شود.
پایتون یک مجموعه قوانین دارد که برای تصمیم‌گیری در مورد اینکه شما به چه متغیرهایی (مانند **x** در این مورد) در کد خود اشاره می‌کنید، آن‌ها را دنبال می‌کند. بیایید قوانین را تجزیه و تحلیل کنیم:

این ایده محدوده در کد شما بسیار مهم است تا بتوانید به درستی نام‌های متغیر را تعیین و فراخوانی کنید.

به زبان ساده، ایده محدوده را می‌توان با ۳ قانون عمومی توصیف کرد:

۱. تخصیص نام به طور پیش فرض نام‌های محلی را ایجاد یا تغییر می‌دهد. ۲. جستجوی نام (حداکثر) در چهار محدوده، این‌ها عبارتند از: * ۳ * Built-in (module) * Global * Enclosing function locals * Local. نام‌های اعلام شده در جملات جهانی و نام‌های غیرمحلی تخصیص داده شده را به محدوده‌های ماژول و تابع محصور نگاشت می‌کنند.

جمله در #۲ بالا را می‌توان با قانون LEGB تعریف کرد.

قانون LEGB:

L: Local — نام‌های تخصیص داده شده به هر نحو در داخل یک تابع (def یا lambda)، که در سطح جهانی اعلام نشده است.

E: Enclosing function locals — نام‌هایی در محدوده محلی هر و همه توابع محصورکننده (def یا لامبدا)، از درونی به بیرونی..

G: Global (module) — نام‌هایی که در سطح بالای یک فایل ماژول اختصاص داده می‌شوند، یا در یک تعریف جهانی در فایل تعریف می‌شوند.

B: Built-in (Python) — نام‌های از پیش تعیین شده در ماژول نام‌های داخلی: SyntaxError، range، open...

مثال های ساده از LEGB

محلی

In [5]:

```
# x is local here:
f = lambda x: x ** 2
```

Enclosing function locals

این حالت زمانی رخ می دهد که ما یک تابع درون یک تابع دیگر داشته باشیم.

In [8]:

```
name = "This is a global name"

def greet():
    # Enclosing function
    name = 'Sammy'

    def hello():
        # name = 'John'  ==> Local
        print('Hello ' + name)
    hello()

greet()
```

```
Hello John
```

توجه کنید که چگونه تابع `hello()` به مقدار متغیر `name` دسترسی پیدا کرد! زیرا تابع `hello()` در داخل تابع `greet()` محصور شده است.

Global

در داخل جوپیتر براحتی می توان جهانی بودن یک متغیر را چک کرد، اگر یک متغیر در سلول دیگری قابل دسترسی باشد پس آن متغیر بصورت جهانی تعریف شده است!

In [9]:

```
name = "This is a global name"

def greet():
    # Enclosing function
    # name = 'Sammy'

    def hello():
        # name = 'John'  ==> Local
        print('Hello ' + name)
    hello()

greet()
```

```
Hello This is a global name
```

In [10]:

```
print(name)
```

```
This is a global name
```

Built-in

برای مثال تابع `len` یا `id` جزو توابع پیش فرض پایتون هستند.

In [11]:

```
print
```

Out[11]:

```
<function print>
```

In [12]:

```
len
```

Out[12]:

```
<function len(obj, /)>
```

In [13]:

```
id
```

Out[13]:

```
<function id(obj, /)>
```

In [15]:

```
# Global scope
# name = "This is a global name"

def greet():
    # Enclosing function scope
    # name = 'Sammy'

    def hello():
        # name = 'John'  ==> Local scope
        print('Hello ' + firstname)
    hello()

greet()
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[15], line 13
     10         print('Hello ' + firstname)
     11     hello()
---> 13 greet()

Cell In[15], line 11, in greet()
      8 def hello():
      9     # name = 'John'  ==> Local scope
     10     print('Hello ' + firstname)
---> 11 hello()

Cell In[15], line 10, in greet.<locals>.hello()
      8 def hello():
      9     # name = 'John'  ==> Local scope
---> 10     print('Hello ' + firstname)

NameError: name 'firstname' is not defined
```

متغیرهای محلی

وقتی متغیرها را در داخل تعریف تابع اعلام می کنید، به هیچ وجه به متغیرهای دیگر با همین نام که خارج از تابع استفاده می شوند، ارتباطی ندارند - به عبارت دیگر، نام متغیرها برای تابع محلی است. این مسئله را دامنه متغیر (Variable scope) نامیده اند. همه متغیرها دامنه بلوکی را که در آن تعریف شده اند، از نقطه تعریف نام خود شروع کرده و دارای دامنه بلوک هستند.

مثال:

In [16]:

```
x = 50

def func(x):
    print('x is', x)
    x = 2
    print('Changed local x to', x)

func(x)
print('x is still', x)
```

```
x is 50
Changed local x to 2
x is still 50
```

اولین باری که مقدار نام **x** را با خط اول در بدنه تابع چاپ می کنیم، پایتون از مقدار پارامتری استفاده می کند که در بلوک اصلی، بالای تعریف تابع تعریف شده است.

سپس، مقدار 2 را به **x** اختصاص می دهیم. نام **x** محلی برای تابع ماست. بنابراین، هنگامی که مقدار **x** را در تابع تغییر می دهیم، **x** در بلوک اصلی تحت تأثیر قرار نمی گیرد.

با آخرین دستور چاپ، مقدار **x** را به عنوان در بلوک اصلی تعریف شده، نمایش می دهیم و به همین ترتیب تأیید می کنیم که در واقع تحت تأثیر قرار نگرفته است.

دستور `global`

اگر می خواهید به یک نام در سطح بالای برنامه (بدون هرگونه دامنه از جمله توابع یا کلاس ها) مقدار اختصاص دهید، باید به پایتون بگوئید که نام مورد نظر شما محلی نبوده و بلکه جهانی است. این کار را با استفاده از دستور `global` انجام می دهیم. اختصاص دادن مقدار به یک متغیر خارج از یک تابع بدون استفاده از دستور global غیرممکن است.

شما می توانید از اینگونه متغیرهای خارج از تابع (با فرض عدم وجود هرگونه متغیر با همین نام در داخل تابع) استفاده کنید. با این حال، این کار پاسخ دادن به خواننده برنامه را سخت و غیر واضح می کند. استفاده از دستور `global` باعث وضوح شود که متغیر در یک بلوک بالاترین سطح تعریف شده است.

مثال:

In [20]:

```
x = 50

def func():
    global x
    print('This function is now using the global x!')
    print('Because of global x is: ', x, id(x))
    x = 2
    print('Ran func(), changed global x to', x, id(x))

print('Before calling func(), x is: ', x, id(x))
func()
print('Value of x (outside of func()) is: ', x, id(x))
```

```
Before calling func(), x is:  50 3032342136592
This function is now using the global x!
Because of global x is:  50 3032342136592
Ran func(), changed global x to 2 3032342135056
Value of x (outside of func()) is:  2 3032342135056
```

دستور `global` برای اعلام این مورد استفاده می شود که **x** یک متغیر جهانی است - بنابراین، هنگامی که مقدار **x** را درون تابع اختصاص می دهیم، آن تغییر در هنگام استفاده از مقدار **x** در بلوک اصلی، نمایش داده می شود.

شما می توانید با استفاده از یک دستور global بیشتر از یک متغیر جهانی را مشخص کنید، به عنوان مثال `global x، y، z` .

نتیجه گیری

شما باید در حال حاضر درک خوبی از Scope داشته باشید (ممکن است قبلاً به طور ذاتی درست درباره Scope حس کرده باشید که عالی است!) چیز دیگری که باید به خاطر داشته باشید این است که همه چیز در پایتون یک شیء است! من می توانم متغیرها را به توابع نسبت دهم مانند کاری که با اعداد انجام میدادیم! ما در بخش decorator دوره به این موضوع دوباره خواهیم پرداخت!