Function Scope, Default Parameters và Lambda Functions

Thời gian: 3 tiếng (09:00-12:00)

Mục tiêu:

- Hiểu về phạm vi (scope) của biến trong functions
- Sử dụng default parameters hiệu quả
- Làm quen với lambda functions cơ bản

Kiến thức cần có:

- Functions cơ bản
- Parameters và return values
- Variables và data types

1. Function Scope (Phạm vi của biến)

1.1 Local Scope vs Global Scope

Local Scope: Biến được tạo bên trong function chỉ có thể sử dụng trong function đó.

In [ ]:
```python
# Ví dụ về Local Scope
def tinh_tong(a, b):
    ket_qua = a + b  # ket_qua là biến local
    print(f"Trong function: {ket_qua}")
    return ket_qua

# Gọi function
tong = tinh_tong(5, 3)
print(f"Ngoài function: {tong}")

# Thử truy cập biến local từ bên ngoài (sẽ lỗi)
# print(ket_qua)  # Uncomment dòng này sẽ báo lỗi
```

Global Scope: Biến được tạo ngoài function có thể sử dụng ở mọi nơi.

In [ ]:
```python
# Ví dụ về Global Scope
ten_truong = "Trường Đại học ABC"  # Biến global

def in_thong_tin_sinh_vien(ten, tuoi):
    print(f"Sinh viên: {ten}")
    print(f"Tuổi: {tuoi}")
    print(f"Trường: {ten_truong}")  # Sử dụng biến global
```

```
in_thong_tin_sinh_vien("Nguyễn Văn A", 20)
```

1.2 Từ khóa global

In [1]:
```
so_lan = 0

def dem_so_lan():
    so_lan += 1

dem_so_lan()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[1], line 6
      3 def dem_so_lan():
      4     so_lan += 1
----> 6 dem_so_lan()

Cell In[1], line 4, in dem_so_lan()
      3 def dem_so_lan():
----> 4     so_lan += 1

UnboundLocalError: local variable 'so_lan' referenced before assignment
```

In [2]:
```
# Sử dụng global để thay đổi biến global trong function
so_dem = 0  # Biến global

def tang_so_dem():
    global so_dem
    so_dem += 1
    print(f"Số đếm hiện tại: {so_dem}")

print(f"Số đếm ban đầu: {so_dem}")
tang_so_dem()
tang_so_dem()
print(f"Số đếm cuối cùng: {so_dem}")
```

```
Số đếm ban đầu: 0
Số đếm hiện tại: 1
Số đếm hiện tại: 2
Số đếm cuối cùng: 2
```

🔥 TODO 1: Thực hành Function Scope

Yêu cầu:

1. Tạo biến global `diem_tong = 0`
2. Viết function `cong_diem(diem)` để cộng điểm vào tổng điểm
3. Viết function `hien_thi_diem()` để hiển thị tổng điểm
4. Test với các giá trị khác nhau

In [3]:
```python
# TODO 1: Viết code ở đây
diem_tong = 0

def cong_diem(diem):
    # Viết code để cộng điểm vào diem_tong
    global diem_tong
    diem_tong += diem
    print(f"Đã cộng {diem} điểm")

def hien_thi_diem():
    # Viết code để hiển thị tổng điểm
    print(f"Tổng điểm hiện tại là {diem_tong}")

# Test functions
hien_thi_diem()
cong_diem(10)
cong_diem(15)
hien_thi_diem()
```

```
Tổng điểm hiện tại là 0
Đã cộng 10 điểm
Đã cộng 15 điểm
Tổng điểm hiện tại là 25
```

2. Default Parameters (Tham số mặc định)

2.1 Cú pháp cơ bản

Default parameters cho phép function hoạt động khi không truyền đủ tham số.

Function không linh hoạt

In [ ]:
```python
def chao_hoi_v1(ten):
    print(f"Xin chào {ten}!")
    print("Bạn 18 tuổi")           # Cố định!
    print("Sống ở Hà Nội")         # Cố định!
```

Thêm tham số

In [ ]:
```python
def chao_hoi_v2(ten, tuoi, thanh_pho):
    print(f"Xin chào {ten}!")
    print(f"Bạn {tuoi} tuổi")
    print(f"Sống ở {thanh_pho}")

# Vấn đề: Phải truyền đủ 3 tham số mãi!
chao_hoi_v2("An", 18, "Hà Nội")
chao_hoi_v2("Bình", 18, "Hà Nội")  # Lặp lại!
```

Default parameters

In [ ]:
```python
def chao_hoi_v3(ten, tuoi=18, thanh_pho="Hà Nội"):
    print(f"Xin chào {ten}!")
    print(f"Bạn {tuoi} tuổi")
    print(f"Sống ở {thanh_pho}")

# Đã linh hoạt!
chao_hoi_v3("An")                    # Dùng default
chao_hoi_v3("Bình", 25)              # Override tuổi
chao_hoi_v3("Cường", 30, "TP.HCM")  # Override tất cả
```

2.2 Keyword Arguments

In [ ]:
```python
# Sử dụng keyword arguments
def tinh_tien_dien(so_kwh, gia_co_ban=3500, phi_dich_vu=50000):
    tien_dien = so_kwh * gia_co_ban
    tong_tien = tien_dien + phi_dich_vu
    return tong_tien

# Các cách gọi function
print(f"Tiền điện 100 kwh: {tinh_tien_dien(100)}")
print(f"Tiền điện 100 kwh (giá 4000): {tinh_tien_dien(100, 4000)}")
print(f"Tiền điện 100 kwh (phí dịch vụ 60000): {tinh_tien_dien(100, phi_dich_vu=600
```

2.3 Lưu ý quan trọng với mutable default values

In [ ]:
```python
# CÁCH SAI - Không nên dùng list làm default value
def them_mon_hoc_sai(mon_hoc, danh_sach=[]):
    danh_sach.append(mon_hoc)
    return danh_sach

# CÁCH ĐÚNG - Dùng None
def them_mon_hoc_dung(mon_hoc, danh_sach=None):
    if danh_sach is None:
        danh_sach = []
    danh_sach.append(mon_hoc)
    return danh_sach

# Test
print("Cách sai:")
print(them_mon_hoc_sai("Toán"))
print(them_mon_hoc_sai("Lý"))  # Sẽ có cả "Toán" và "Lý"

print("\nCách đúng:")
print(them_mon_hoc_dung("Toán"))
print(them_mon_hoc_dung("Lý"))  # Chỉ có "Lý"
```

In [5]:
```python
def them_san_pham_vao_gio_hang(san_pham, gio_hang = []):
    gio_hang.append(san_pham)
    print(f"Đã thêm {san_pham} sản phẩm vào giỏ hàng")
    return gio_hang

print("\n 👤 Khách hàng A: ")
gio_hang_A = them_san_pham_vao_gio_hang("Bánh mì")
```

```
print(f"Giỏ hàng A: {gio_hang_A}")


print("\n 👤 Khách hàng B: ")
gio_hang_B = them_san_pham_vao_gio_hang("Quần áo")
print(f"Giỏ hàng B: {gio_hang_B}")

print("\n Check lại giỏ hàng của khách hàng A")
print(f"Giỏ hàng A: {gio_hang_A}")

print("\n Check lại giỏ hàng của khách hàng B")
print(f"Giỏ hàng B: {gio_hang_B}")
```

```
 Khách hàng A:
Đã thêm Bánh mì sản phẩm vào giỏ hàng
Giỏ hàng A: ['Bánh mì']

 Khách hàng B:
Đã thêm Quần áo sản phẩm vào giỏ hàng
Giỏ hàng B: ['Bánh mì', 'Quần áo']

 Check lại giỏ hàng của khách hàng A
Giỏ hàng A: ['Bánh mì', 'Quần áo']

 Check lại giỏ hàng của khách hàng B
Giỏ hàng B: ['Bánh mì', 'Quần áo']
```

🔥 TODO 2: Thực hành Default Parameters

Yêu cầu:

1. Viết function `tinh_luong(luong_co_ban, thuong=0, bao_hiem=10)`
2. Tính lương = lương cơ bản + thưởng - (lương cơ bản * bảo hiểm/100)
3. Test với các trường hợp khác nhau

In [ ]:
```
# TODO 2: Viết code ở đây
def tinh_luong(luong_co_ban, thuong=0, bao_hiem=10):
    # Viết code tính lương
    tien_bao_hiem = luong_co_ban * bao_hiem / 100
    luong_thuc_nhan = luong_co_ban + thuong - tien_bao_hiem
    return luong_thuc_nhan

# Test cases
print(f"Lương 10tr: {tinh_luong(10000000)}")
print(f"Lương 10tr + thưởng 2tr: {tinh_luong(10000000, 2000000)}")
print(f"Lương 10tr + thưởng 2tr + bảo hiểm 15%: {tinh_luong(10000000, 2000000, 15)}")
```

3. Lambda Functions (Hàm ẩn danh)

3.1 Cú pháp cơ bản

Lambda functions là các hàm nhỏ, chỉ có một biểu thức.

In [1]:
```python
# So sánh function thường và lambda

# Function thường
def binh_phuong(x):
    return x ** 2

# Lambda function
binh_phuong_lambda = lambda x: x ** 2

# Test
print(f"Function thường: {binh_phuong(5)}")
print(f"Lambda function: {binh_phuong_lambda(5)}")
```

```
Function thường: 25
Lambda function: 25
```

3.2 Lambda với nhiều tham số

In [6]:
```python
# Lambda với nhiều tham số
cong = lambda a, b: a + b
nhan = lambda a, b: a * b
tim_max = lambda a, b, c: max(a, b, c)

# Test
print(f"3 + 5 = {cong(3, 5)}")
print(f"4 * 6 = {nhan(4, 6)}")
print(f"Max(10, 5, 8) = {tim_max(10, 5, 8)}")
```

```
3 + 5 = 8
4 * 6 = 24
Max(10, 5, 8) = 10
```

3.3 Sử dụng Lambda với map(), filter(), sorted()

In [8]:
```python
danh_sach_vi_du = [1, 2, 3, 4]
danh_sach_binh_phuong = []
for danh_sach in danh_sach_vi_du:
    danh_sach_binh_phuong.append(danh_sach ** 2)

danh_sach_binh_phuong
```

Out[8]:
```
[1, 4, 9, 16]
```

In [7]:
```python
# Lambda với map() - áp dụng function cho mỗi phần tử
danh_sach_so = [1, 2, 3, 4, 5]

# Bình phương từng số
binh_phuong_list = list(map(lambda x: x**2, danh_sach_so))
print(f"Bình phương: {binh_phuong_list}")

# Nhân đôi từng số
nhan_doi_list = list(map(lambda x: x*2, danh_sach_so))
print(f"Nhân đôi: {nhan_doi_list}")
```

```
Bình phương: [1, 4, 9, 16, 25]
Nhân đôi: [2, 4, 6, 8, 10]
```

In [1]:
```python
# Lambda với filter() - lọc phần tử
danh_sach_so = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Lọc số chẵn
so_chan = list(filter(lambda x: x % 2 == 0, danh_sach_so))
print(f"Số chẵn: {so_chan}")

# Lọc số lớn hơn 5
so_lon = list(filter(lambda x: x > 5, danh_sach_so))
print(f"Số > 5: {so_lon}")
```

```
Số chẵn: [2, 4, 6, 8, 10]
Số > 5: [6, 7, 8, 9, 10]
```

In [2]:
```python
# Lambda với sorted() - sắp xếp
sinh_vien = [
    {"ten": "An", "diem": 8.5},
    {"ten": "Bình", "diem": 7.0},
    {"ten": "Cường", "diem": 9.5},
    {"ten": "Dung", "diem": 6.5}
]

# Sắp xếp theo điểm
sap_xep_diem = sorted(sinh_vien, key=lambda x: x["diem"])
print("Sắp xếp theo điểm:")
for sv in sap_xep_diem:
    print(f"  {sv['ten']}: {sv['diem']}")

# Sắp xếp theo tên
sap_xep_ten = sorted(sinh_vien, key=lambda x: x["ten"])
print("\nSắp xếp theo tên:")
for sv in sap_xep_ten:
    print(f"  {sv['ten']}: {sv['diem']}")
```

```
Sắp xếp theo điểm:
  Dung: 6.5
  Bình: 7.0
  An: 8.5
  Cường: 9.5

Sắp xếp theo tên:
  An: 8.5
  Bình: 7.0
  Cường: 9.5
  Dung: 6.5
```

🔥 TODO 3: Thực hành Lambda Functions

Yêu cầu:

1. Tạo lambda function `chuyen_c_sang_f` để chuyển độ C sang độ F
2. Tạo lambda function `tinh_dien_tich_hinh_chu_nhat`

3. Sử dụng lambda với map() để chuyển list nhiệt độ C sang F
4. Sử dụng lambda với filter() để lọc các số chia hết cho 3

In [ ]:
```python
# TODO 3: Viết code ở đây

# 1. Lambda chuyển C sang F (F = C * 9/5 + 32)
chuyen_c_sang_f = lambda c: # Hoàn thành

# 2. Lambda tính diện tích hình chữ nhật
tinh_dien_tich_hinh_chu_nhat = lambda dai, rong: # Hoàn thành

# 3. Chuyển list nhiệt độ C sang F
nhiet_do_c = [0, 10, 20, 30, 40]
nhiet_do_f = # Sử dụng map() và lambda

# 4. Lọc số chia hết cho 3
danh_sach_so = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
chia_het_cho_3 = # Sử dụng filter() và lambda

# Test
print(f"25°C = {chuyen_c_sang_f(25)}°F")
print(f"Diện tích 5x3 = {tinh_dien_tich_hinh_chu_nhat(5, 3)}")
print(f"Nhiệt độ F: {nhiet_do_f}")
print(f"Chia hết cho 3: {chia_het_cho_3}")
```

4. Bài tập tổng hợp

🔥 TODO 4: Xây dựng hệ thống quản lý điểm số

Yêu cầu:

1. Tạo biến global để lưu danh sách sinh viên
2. Viết function `them_sinh_vien(ten, diem_toan=0, diem_ly=0, diem_hoa=0)`
3. Viết function `tinh_diem_trung_binh()` sử dụng lambda
4. Viết function `loc_sinh_vien_gioi()` (điểm TB >= 8.0)
5. Viết function `sap_xep_theo_diem_tb()`

In [ ]:
```python
# TODO 4: Viết code ở đây

# 1. Biến global
danh_sach_sinh_vien = []

# 2. Function thêm sinh viên
def them_sinh_vien(ten, diem_toan=0, diem_ly=0, diem_hoa=0):
    # Viết code
    pass

# 3. Function tính điểm trung bình
def tinh_diem_trung_binh():
    # Sử dụng lambda để tính TB cho mỗi sinh viên
    pass
```

```
# 4. Function lọc sinh viên giỏi
def loc_sinh_vien_gioi():
    # Sử dụng filter và lambda
    pass

# 5. Function sắp xếp theo điểm TB
def sap_xep_theo_diem_tb():
    # Sử dụng sorted và lambda
    pass

# Test
them_sinh_vien("An", 8, 7, 9)
them_sinh_vien("Bình", 6, 8, 7)
them_sinh_vien("Cường", 9, 9, 8)
them_sinh_vien("Dung", 5, 6, 7)

print("Danh sách sinh viên:")
tinh_diem_trung_binh()

print("\nSinh viên giỏi:")
sinh_vien_gioi = loc_sinh_vien_gioi()
for sv in sinh_vien_gioi:
    print(f"  {sv}")

print("\nSắp xếp theo điểm TB:")
sap_xep_theo_diem_tb()
```

5. Tổng kết

Kiến thức đã học:

1. **Function Scope:**
   - Local scope vs Global scope
   - Từ khóa `global`
   - Cách truy cập biến trong các phạm vi khác nhau
2. **Default Parameters:**
   - Cú pháp và cách sử dụng
   - Keyword arguments
   - Lưu ý với mutable default values
3. **Lambda Functions:**
   - Cú pháp cơ bản
   - Sử dụng với map(), filter(), sorted()
   - Khi nào nên dùng lambda

Buổi học tiếp theo:

- File handling cơ bản
- Reading/writing files
- CSV basics

6. Bài tập về nhà

🏠 Bài tập 1: Quản lý kho hàng

Viết chương trình quản lý kho hàng với các yêu cầu:

- Biến global lưu danh sách sản phẩm
- Function thêm sản phẩm với default parameters
- Sử dụng lambda để tính tổng giá trị kho
- Lọc sản phẩm hết hàng

🏠 Bài tập 2: Máy tính nâng cao

Tạo máy tính sử dụng lambda functions cho các phép tính:

- Cộng, trừ, nhân, chia
- Tính lũy thừa, căn bậc hai
- Cho phép người dùng chọn phép tính

🏠 Bài tập 3: Phân tích dữ liệu đơn giản

Cho danh sách điểm thi của học sinh:

- Tính điểm trung bình
- Lọc học sinh đậu (>= 5.0)
- Sắp xếp theo điểm
- Tìm điểm cao nhất, thấp nhất