

# Chương 1

## Hàm Cơ Bản và Hàm Tự Tạo

### 1.1 Giới Thiệu Về Hàm

#### Lý Thuyết

##### Hàm là gì?

Hàm (Function) là một khối code được đặt tên, có thể được gọi nhiều lần để thực hiện một tác vụ cụ thể. Hàm giúp:

- **Tái sử dụng code:** Viết một lần, dùng nhiều lần
- **Tổ chức code:** Chia nhỏ chương trình thành các phần logic
- **Dễ bảo trì:** Sửa lỗi chỉ cần sửa ở một chỗ
- **Tăng tính đọc hiểu:** Code rõ ràng, dễ hiểu hơn

#### Phân loại hàm trong Python:

- **Hàm built-in:** Có sẵn trong Python (print, input, len, max...)
- **Hàm từ thư viện:** Import từ module (math.sqrt, random.choice...)
- **Hàm tự định nghĩa:** Người dùng tạo ra bằng từ khóa `def`

### 1.2 Hàm Built-in Thường Dùng

#### Lý Thuyết

##### Các hàm built-in cơ bản:

###### 1. Hàm Input/Output:

- `print()`: In ra màn hình
- `input()`: Nhập từ bàn phím

###### 2. Hàm chuyển đổi kiểu dữ liệu:

- `int()`: Chuyển sang số nguyên
- `float()`: Chuyển sang số thực
- `str()`: Chuyển sang chuỗi
- `bool()`: Chuyển sang boolean

### 3. Hàm toán học:

- `abs()`: Giá trị tuyệt đối
- `max()`: Tìm giá trị lớn nhất
- `min()`: Tìm giá trị nhỏ nhất
- `sum()`: Tính tổng
- `round()`: Làm tròn
- `pow()`: Lũy thừa

### 4. Hàm xử lý dữ liệu:

- `len()`: Độ dài chuỗi/danh sách
- `type()`: Kiểu dữ liệu
- `range()`: Tạo dãy số
- `sorted()`: Sắp xếp

## Code Mẫu

```

1 # Vi du su dung cac ham built-in
2 print("==== CAC HAM BUILT-IN ====")
3
4 # Ham Input/Output
5 ten = input("Nhap ten: ")
6 print(f"Xin chao {ten}!")
7
8 # Ham chuyen doi kieu du lieu
9 so_str = "123"
10 so_int = int(so_str)
11 so_float = float(so_str)
12 print(f"String: {so_str}, Int: {so_int}, Float: {so_float}")
13
14 # Ham toan hoc
15 numbers = [3, 7, 1, 9, 5]
16 print(f"Danh sach: {numbers}")
17 print(f"Max: {max(numbers)}")
18 print(f"Min: {min(numbers)}")

```

```
19 print(f"Sum: {sum(numbers)}")
20 print(f"Abs(-5): {abs(-5)}")
21 print(f"Round(3.14159, 2): {round(3.14159, 2)}")
22 print(f"Pow(2, 3): {pow(2, 3)}")
23
24 # Ham xu ly du lieu
25 text = "Hello Python"
26 print(f"Do dai chuoi '{text}': {len(text)}")
27 print(f"Kieu du lieu: {type(text)}")
28 print(f"Sap xep: {sorted(numbers)}")
```

## 1.3 Hàm Từ Thư Viện

### Lý Thuyết

#### Thư viện math:

- `math.sqrt()`: Căn bậc hai
- `math.ceil()`: Làm tròn lên
- `math.floor()`: Làm tròn xuống
- `math.pi`: Hằng số  $\pi$
- `math.e`: Hằng số e
- `math.sin()`, `math.cos()`, `math.tan()`: Hàm lượng giác

#### Thư viện random:

- `random.random()`: Số ngẫu nhiên từ 0-1
- `random.randint(a, b)`: Số nguyên ngẫu nhiên từ a đến b
- `random.choice()`: Chọn ngẫu nhiên từ danh sách
- `random.shuffle()`: Trộn danh sách

#### Thư viện datetime:

- `datetime.now()`: Thời gian hiện tại
- `datetime.date()`: Ngày tháng
- `datetime.time()`: Giờ phút giây

### Code Mẫu

```

1 import math
2 import random
3 from datetime import datetime
4
5 print("==== HAM TU THU VIEN ====")
6
7 # Thu vien math
8 print(f".Sqrt(16): {math.sqrt(16)}")
9 print(f".Ceil(3.2): {math.ceil(3.2)}")
10 print(f".Floor(3.8): {math.floor(3.8)}")
11 print(f".Pi: {math.pi:.4f}")
12 print(f".Sin(30 do): {math.sin(math.radians(30)):.4f}")
13
14 # Thu vien random
15 print(f".Random 0-1: {random.random():.4f}")
16 print(f".Randint 1-10: {random.randint(1, 10)}")
17 colors = ["do", "xanh", "vang", "tim"]
18 print(f".Choice color: {random.choice(colors)}")
19
20 # Thu vien datetime
21 now = datetime.now()
22 print(f".Thoi gian hien tai: {now.strftime('%d/%m/%Y %H:%M:%S')}")

```

## 1.4 Định Nghĩa Hàm Tự Tạo

### Lý Thuyết

#### Cú pháp định nghĩa hàm:

```

1 def ten_ham(tham_so1, tham_so2, ...):
2     """Docstring - Mô tả hàm"""
3     # Than ham
4     # Xu ly logic
5     return ket_qua # Tuy chon

```

#### Các thành phần của hàm:

- **def:** Từ khóa định nghĩa hàm
- **ten\_ham:** Tên hàm (tuân theo quy tắc đặt tên biến)
- **tham\_so:** Đầu vào của hàm (parameters)
- **docstring:** Mô tả chức năng hàm (tùy chọn)
- **return:** Trả về kết quả (tùy chọn)

Gọi hàm:

```
1 ket_qua = ten_ham(gia_tril, gia_tril2, ...)
```

### Code Mẫu

```

1 # Vi du ham don gian
2 def chao_hoi():
3     """Ham chao hoi don gian"""
4     print("Xin chao! Welcome to Python!")
5
6 def chao_hoi_ca_nhan(ten):
7     """Ham chao hoi voi ten"""
8     print(f"Xin chao {ten}! Rat vui duoc gap ban!")
9
10 def tinh_tong(a, b):
11     """Ham tinh tong hai so"""
12     ket_qua = a + b
13     return ket_qua
14
15 def tinh_dien_tich_hinh_vuong(canh):
16     """Ham tinh dien tich hinh vuong"""
17     dien_tich = canh * canh
18     return dien_tich
19
20 # Gọi ham
21 print("== SU DUNG HAM TU TAO ==")
22 chao_hoi()
23 chao_hoi_ca_nhan("An")
24
25 tong = tinh_tong(5, 3)
26 print(f"5 + 3 = {tong}")
27
28 dt = tinh_dien_tich_hinh_vuong(4)
29 print(f"Dien tich hinh vuong canh 4: {dt}")

```

## 1.5 Tham Số và Đối Số

### Lý Thuyết

**Phân loại tham số:**

1. Tham số bắt buộc (Required Parameters):

```
1 def tinh_hieu(a, b):
2     return a - b
```

2. Tham số mặc định (Default Parameters):

```

1 def chao(ten, ngon_ngu="Tieng Viet"):
2     return f"Xin chao {ten} bang {ngon_ngu}!"
```

### 3. Tham số từ khóa (Keyword Arguments):

```

1 def thong_tin(ten, tuoi, lop):
2     return f"{ten}, {tuoi} tuoi, lop {lop}"
3
4 # Gọi ham voi keyword arguments
5 print(thong_tin(tuoi=16, lop="10A", ten="An"))
```

### 4. Tham số biến đổi (\*args, \*\*kwargs):

```

1 def tinh_tong_nhieu_so(*numbers):
2     return sum(numbers)
3
4 def in_thong_tin(**info):
5     for key, value in info.items():
6         print(f"{key}: {value}")
```

## Code Mẫu

```

1 # Vi du chi tiet ve tham so
2 def tinh_toan(a, b, phep_toan="+", in_ket_qua=True):
3     """
4     Ham tinh toan voi nhieu tham so
5     a, b: cac so (bat buoc)
6     phep_toan: phep tinh (mac dinh la "+")
7     in_ket_qua: co in ket qua khong (mac dinh True)
8     """
9
10    if phep_toan == "+":
11        ket_qua = a + b
12    elif phep_toan == "-":
13        ket_qua = a - b
14    elif phep_toan == "*":
15        ket_qua = a * b
16    elif phep_toan == "/":
17        if b != 0:
18            ket_qua = a / b
19        else:
20            return "Loi: Khong the chia cho 0"
21    else:
22        return "Phep toan khong hop le"
23
24    if in_ket_qua:
25        print(f"[{a} {phep_toan} {b}] = {ket_qua}")
26
27    return ket_qua
```

```

28 # Su dung ham voi cac cach khac nhau
29 print("== THAM SO VA DOI SO ==")
30 tinh_toan(10, 5)                      # Mac dinh: +, in ket qua
31 tinh_toan(10, 5, "*")                  # Nhan, in ket qua
32 tinh_toan(10, 5, "/", False)          # Chia, khong in
33 result = tinh_toan(phep_toan="-", a=15, b=7)  # Keyword
34     arguments
35
36 # Ham voi *args
37 def tinh_tong_nhieu(*numbers):
38     """Tinh tong nhieu so"""
39     return sum(numbers)
40
41
42 # Ham voi **kwargs
43 def in_profile(**thong_tin):
44     """In thong tin ca nhan"""
45     print("--- THONG TIN CA NHAN ---")
46     for key, value in thong_tin.items():
47         print(f"{key.title()}: {value}")
48
49 in_profile(ten="Nguyen Van A", tuoi=16, lop="10A",
so_thich="Lap trinh")

```

## 1.6 Phạm Vi Biến (Variable Scope)

### Lý Thuyết

Các loại phạm vi biến:

#### 1. Biến toàn cục (Global Variable):

- Được định nghĩa bên ngoài hàm
- Có thể truy cập từ mọi nơi trong chương trình
- Sử dụng từ khóa global để sửa đổi trong hàm

#### 2. Biến cục bộ (Local Variable):

- Được định nghĩa bên trong hàm
- Chỉ tồn tại trong phạm vi hàm đó
- Bị hủy khi hàm kết thúc

### Quy tắc LEGB:

- Local: Biến cục bộ (trong hàm)

- Enclosing: Biến trong hàm bao ngoài
- Global: Biến toàn cục
- Built-in: Biến built-in của Python

### Code Mẫu

```

1 # Vi du ve pham vi bien
2 # Bien toan cuc
3 counter = 0
4 pi = 3.14159
5
6 def tang_counter():
7     """Ham tang bien counter toan cuc"""
8     global counter
9     counter += 1
10    print(f"Counter: {counter}")
11
12 def tinh_dien_tich_tron(ban_kinh):
13     """Ham tinh dien tich hinh tron"""
14     # Bien cuc bo
15     dien_tich = pi * ban_kinh * ban_kinh
16     return dien_tich
17
18 def demo_scope():
19     """Demo pham vi bien"""
20     # Bien cuc bo
21     local_var = "Toi la bien cuc bo"
22     print(local_var)
23     print(f"Truy cap bien toan cuc pi: {pi}")
24
25     # Khong the truy cap local_var o ben ngoai ham nay
26
27 print("== PHAM VI BIEN ==")
28 print(f"Counter ban dau: {counter}")
29 tang_counter()
30 tang_counter()
31
32 dt = tinh_dien_tich_tron(5)
33 print(f"Dien tich hinh tron ban kinh 5: {dt:.2f}")
34
35 demo_scope()
36 # print(local_var) # Loi! Khong the truy cap bien cuc bo
37
38 # Ham long nhau (Nested Functions)
39 def ham ngoai():
40     """Ham ngoai chua ham trong"""
41     bien_ngoai = "Bien trong ham ngoai"
42

```

```

13     def ham_trong():
14         """Ham trong"""
15         bien_trong = "Bien trong ham trong"
16         print(f"Trong ham trong: {bien ngoai}") # Truy cap
17             bien enclosing
18         print(f"Trong ham trong: {bien_trong}")
19
20     ham_trong()
21     print(f"Trong ham ngoai: {bien ngoai}")
22     # print(bien_trong) # Loi! Khong truy cap duoc
23
24 ham ngoai()

```

## 1.7 Bài Tập Về Hàm (Bài 56-65)

### Bài Tập

#### Bài 56: Hàm kiểm tra số chẵn lẻ

Viết hàm `kiem_tra_chan_le(n)` kiểm tra một số là chẵn hay lẻ.

**Input:** Số nguyên n

**Output:** "Chan" hoặc "Le"

**Ví dụ:**

`kiem_tra_chan_le(6) → "Chan"`  
`kiem_tra_chan_le(7) → "Le"`

### Hướng Dẫn

#### Hướng dẫn:

1. Định nghĩa hàm với tham số n
2. Kiểm tra  $n \% 2 == 0$
3. Return "Chan" hoặc "Le"

### Code Mẫu

```

1 def kiem_tra_chan_le(n):
2     """
3     Kiem tra so chan le
4     Input: n (int) - so nguyen can kiem tra
5     Output: str - "Chan" hoac "Le"
6     """
7     if n % 2 == 0:
8         return "Chan"
9     else:
10        return "Le"

```

```

11
12 # Test ham
13 print("==== BAI 56: KIEM TRA CHAN LE ===")
14 test_numbers = [6, 7, 0, -4, -3]
15 for num in test_numbers:
16     result = kiem_tra_chan_le(num)
17     print(f"{num} la so {result}")

```

## Bài Tập

### Bài 57: Hàm tính giai thừa

Viết hàm `giai_thua(n)` tính giai thừa của số nguyên dương  $n$ .

**Input:** Số nguyên dương  $n$

**Output:**  $n! = 1 \times 2 \times 3 \times \dots \times n$

**Ví dụ:**

`giai_thua(5) → 120`

`giai_thua(0) → 1`

## Hướng Dẫn

### Hướng dẫn:

1. Xử lý trường hợp đặc biệt:  $n = 0$  hoặc  $n = 1 \rightarrow \text{return } 1$
2. Dùng vòng lặp từ 1 đến  $n$
3. Nhân dần các số lại

## Code Mẫu

```

1 def giai_thua(n):
2     """
3         Tinh giai thua cua n
4         Input: n (int) - so nguyen duong
5         Output: int - gia tri giai thua
6         """
7
8     if n < 0:
9         return "Loi: Giai thua khong xac dinh cho so am"
10
11    if n == 0 or n == 1:
12        return 1
13
14    ket_qua = 1
15    for i in range(2, n + 1):
16        ket_qua *= i
17
18    return ket_qua

```

```

19 # Test ham
20 print("== BAI 57: GIAI THUA ==")
21 test_numbers = [0, 1, 5, 7, -3]
22 for num in test_numbers:
23     result = gaii_thua(num)
24     print(f"{num}! = {result}")
25
26 # Cach khac: Su dung de quy
27 def gaii_thua_de_quy(n):
28     """Tinh gaii thua bang de quy"""
29     if n < 0:
30         return "Loi: Khong xac dinh"
31     if n == 0 or n == 1:
32         return 1
33     return n * gaii_thua_de_quy(n - 1)
34
35 print("\nSu dung de quy:")
36 print(f"5! = {gaii_thua_de_quy(5)}")

```

## Bài Tập

### Bài 58: Hàm kiểm tra số nguyên tố

Viết hàm `kiem_tra_nguyen_to(n)` kiểm tra một số có phải số nguyên tố không.

**Input:** Số nguyên dương n

**Output:** True nếu là số nguyên tố, False nếu không

**Ví dụ:**

`kiem_tra_nguyen_to(17) → True`

`kiem_tra_nguyen_to(15) → False`

## Hướng Dẫn

### Hướng dẫn:

1. Số nguyên tố  $> 1$  và chỉ chia hết cho 1 và chính nó
2. Kiểm tra từ 2 đến  $\sqrt{n}$
3. Tối ưu: chỉ cần kiểm tra đến căn bậc hai của n

## Code Mẫu

```

1 import math
2
3 def kiem_tra_nguyen_to(n):
4     """
5         Kiem tra so nguyen to
6         Input: n (int) - so nguyen duong
7         Output: bool - True neu la so nguyen to

```

```

8     """
9     if n < 2:
10        return False
11
12    if n == 2:
13        return True
14
15    if n % 2 == 0:
16        return False
17
18    # Kiem tra cac so le tu 3 den sqrt(n)
19    for i in range(3, int(math.sqrt(n)) + 1, 2):
20        if n % i == 0:
21            return False
22
23    return True
24
25 # Test ham
26 print("== BAI 58: KIEM TRA NGUYEN TO ==")
27 test_numbers = [2, 3, 4, 15, 17, 25, 29, 100, 101]
28 for num in test_numbers:
29     result = kiem_tra_nguyen_to(num)
30     status = "la" if result else "khong phai"
31     print(f"{num} {status} so nguyen to")
32
33 # Ham bo tro: Tim tat ca so nguyen to tu 1 den n
34 def tim_nguyen_to_den_n(n):
35     """Tim tat ca so nguyen to tu 2 den n"""
36     nguyen_to = []
37     for i in range(2, n + 1):
38         if kiem_tra_nguyen_to(i):
39             nguyen_to.append(i)
40     return nguyen_to
41
42 print(f"\nCac so nguyen to tu 1 den 30:
{tim_nguyen_to_den_n(30)}")

```

## Bài Tập

### Bài 59: Hàm tính diện tích hình học

Viết các hàm tính diện tích:

- dien\_tich\_hinh\_vuong(canh)
- dien\_tich\_hinh\_chu\_nhat(dai, rong)
- dien\_tich\_hinh\_tron(ban\_kinh)
- dien\_tich\_tam\_giac(a, b, c) - dùng công thức Heron

**Ví dụ:**

dien\_tich\_hinh\_vuong(5) → 25  
dien\_tich\_hinh\_tron(3) → 28.27

**Hướng Dẫn****Hướng dẫn:**

1. Mỗi hàm tính một loại hình cụ thể
2. Sử dụng các công thức toán học phù hợp
3. Xử lý trường hợp đầu vào không hợp lệ

**Code Mẫu**

```
1 import math
2
3 def dien_tich_hinh_vuong(canh):
4     """
5         Tinh dien tich hinh vuong
6         Input: canh (float) - do dai canh
7         Output: float - dien tich
8     """
9     if canh <= 0:
10         return "Loi: Canh phai duong"
11     return canh * canh
12
13 def dien_tich_hinh_chu_nhat(dai, rong):
14     """
15         Tinh dien tich hinh chu nhat
16         Input: dai, rong (float) - chieu dai va chieu rong
17         Output: float - dien tich
18     """
19     if dai <= 0 or rong <= 0:
20         return "Loi: Cac canh phai duong"
21     return dai * rong
22
23 def dien_tich_hinh_tron(ban_kinh):
24     """
25         Tinh dien tich hinh tron
26         Input: ban_kinh (float) - ban kinh
27         Output: float - dien tich
28     """
29     if ban_kinh <= 0:
30         return "Loi: Ban kinh phai duong"
31     return math.pi * ban_kinh * ban_kinh
32
33 def dien_tich_tam_giac(a, b, c):
34     """
```

```

35     Tinh dien tich tam giac bang cong thuc Heron
36     Input: a, b, c (float) - ba canh tam giac
37     Output: float - dien tich
38     """
39
40     # Kiem tra tam giac hop le
41     if a <= 0 or b <= 0 or c <= 0:
42         return "Loi: Cac canh phai duong"
43
44     if a + b <= c or b + c <= a or a + c <= b:
45         return "Loi: Khong tao thanh tam giac"
46
47     # Cong thuc Heron
48     s = (a + b + c) / 2
49     dien_tich = math.sqrt(s * (s - a) * (s - b) * (s - c))
50     return dien_tich
51
52     # Test cac ham
53     print("==== BAI 59: DIEN TICH HINH HOC ===")
54
55     print("Hinh vuong canh 5:", dien_tich_hinh_vuong(5))
56     print("Hinh chu nhat 6x4:", dien_tich_hinh_chu_nhat(6, 4))
57     print("Hinh tron ban kinh 3:",
58          f"{dien_tich_hinh_tron(3):.2f}")
59     print("Tam giac canh 3,4,5:", dien_tich_tam_giac(3, 4, 5))
60
61     # Test truong hop loi
62     print("\nTest loi:")
63     print("Hinh vuong canh -2:", dien_tich_hinh_vuong(-2))
64     print("Tam giac canh 1,2,5:", dien_tich_tam_giac(1, 2, 5))
65
66     # Ham tong hop tinh dien tich
67     def tinh_dien_tich(hinh, **kwargs):
68         """
69             Ham tong hop tinh dien tich
70             Input: hinh (str), **kwargs - cac tham so theo hinh
71             """
72
73         if hinh == "vuong":
74             return dien_tich_hinh_vuong(kwargs.get('canh', 0))
75         elif hinh == "chu_nhat":
76             return dien_tich_hinh_chu_nhat(kwargs.get('dai', 0),
77                                            kwargs.get('rong', 0))
78         elif hinh == "tron":
79             return dien_tich_hinh_tron(kwargs.get('ban_kinh', 0))
80         elif hinh == "tam_giac":
81             return dien_tich_tam_giac(kwargs.get('a', 0),
82                                      kwargs.get('b', 0), kwargs.get('c', 0))
83         else:
84             return "Loi: Hinh khong hop le"
85
86     print("\nSu dung ham tong hop:")

```

```

82 print("Vuong:", tinh_dien_tich("vuong", canh=4))
83 print("Tron:", f"{tinh_dien_tich('tron', ban_kinh=2):.2f}")

```

## Bài Tập

### Bài 60: Hàm xử lý chuỗi

Viết các hàm xử lý chuỗi:

- `dem_tu(chuoi)`: Đếm số từ trong chuỗi
- `dao_nguoc_chuoi(chuoi)`: Đảo ngược chuỗi
- `kiem_tra_palindrome(chuoi)`: Kiểm tra chuỗi đối xứng
- `chuan_hoa_ten(ten)`: Chuẩn hóa tên (viết hoa chữ cái đầu)

Ví dụ:

`dem_tu("Hello Python World")` → 3

`kiem_tra_palindrome("madam")` → True

## Hướng Dẫn

### Hướng dẫn:

1. Sử dụng các phương thức chuỗi: `split()`, `strip()`, `lower()`, `upper()`
2. Xử lý khoảng trắng và ký tự đặc biệt
3. Kiểm tra chuỗi rỗng và None

## Code Mẫu

```

1 def dem_tu(chuoi):
2     """
3         Dem so tu trong chuoi
4         Input: chuoi (str)
5         Output: int - so luong tu
6     """
7     if not chuoi or not chuoi.strip():
8         return 0
9
10    # Tach chuoi thanh cac tu
11    cac_tu = chuoi.strip().split()
12    return len(cac_tu)
13
14 def dao_nguoc_chuoi(chuoi):
15     """
16         Dao nguoc chuoi
17         Input: chuoi (str)

```

```
18     Output: str - chuoi da dao nguoc
19     """
20     return chuoi[::-1]
21
22 def kiem_tra_palindrome(chuoi):
23     """
24         Kiem tra chuoi doi xung (palindrome)
25         Input: chuoi (str)
26         Output: bool - True neu la palindrome
27         """
28     # Chuyen ve chu thuong va bo khoang trang
29     chuoi_sach = chuoi.lower().replace(" ", "")
30     return chuoi_sach == chuoi_sach[::-1]
31
32 def chuan_hoa_ten(ten):
33     """
34         Chuan hoa ten (viet hoa chu cai dau moi tu)
35         Input: ten (str)
36         Output: str - ten da chuan hoa
37         """
38     if not ten:
39         return ""
40
41     # Tach thanh cac tu va chuan hoa
42     cac_tu = ten.strip().split()
43     cac_tu_chuan_hoa = [tu.capitalize() for tu in cac_tu]
44     return " ".join(cac_tu_chuan_hoa)
45
46 def thong_ke_ky_tu(chuoi):
47     """
48         Thong ke cac ky tu trong chuoi
49         Input: chuoi (str)
50         Output: dict - thong ke tan suat
51         """
52     thong_ke = {}
53     for ky_tu in chuoi.lower():
54         if ky_tu.isalpha(): # Chi dem chu cai
55             thong_ke[ky_tu] = thong_ke.get(ky_tu, 0) + 1
56     return thong_ke
57
58 # Test cac ham
59 print("== BAI 60: XU LY CHUOI ==")
60
61 test_string = "Hello Python World"
62 print(f'{test_string} co {dem_tu(test_string)} tu")
63
64 reverse_str = dao_nguoc_chuoi("Python")
65 print(f'Dao nguoc 'Python': '{reverse_str}'")
66
67 palindromes = ["madam", "racecar", "hello", "A man a plan a
```

```

        canal Panama"]
68 for p in palindromes:
69     result = kiem_tra_palindrome(p)
70     print(f'{p} {\'la\' if result else \'khong phai\'}
71         palindrome')
72
72 names = ["nguyen van an", "TRAN THI BINH", " le minh
73     duc "]
73 for name in names:
74     print(f'{name} -> {chuan_hoa_ten(name)}')
75
76 # Thong ke ky tu
77 text = "Hello World"
78 stats = thong_ke_ky_tu(text)
79 print(f'\nThong ke ky tu trong \'text\':')
80 for char, count in sorted(stats.items()):
81     print(f" {char}: {count}")

```

## Bài Tập

### Bài 61: Hàm tìm kiếm và sắp xếp

Viết các hàm:

- `tim_max_min(danh_sach)`: Tìm giá trị lớn nhất và nhỏ nhất
- `tim_kiem_tuyen_tinh(danh_sach, gia_tri)`: Tìm vị trí của giá trị
- `sap_xep_noi_bot(danh_sach)`: Sắp xếp nổi bợt
- `sap_xep_chon(danh_sach)`: Sắp xếp chọn

Ví dụ:

`tim_max_min([3, 7, 1, 9, 5])` → (9, 1)  
`tim_kiem_tuyen_tinh([1, 3, 5, 7], 5)` → 2

## Hướng Dẫn

### Hướng dẫn:

1. Kiểm tra danh sách rỗng trước khi xử lý
2. Tìm kiếm tuyến tính: duyệt từ đầu đến cuối
3. Sắp xếp nổi bợt: so sánh cặp phần tử liền kề
4. Sắp xếp chọn: tìm phần tử nhỏ nhất, đưa về đầu

## Code Mẫu

```

1 def tim_max_min(danh_sach):
2     """
3         Tim gia tri lon nhat va nho nhat
4         Input: danh_sach (list)
5         Output: tuple (max, min)
6         """
7     if not danh_sach:
8         return None, None
9
10    max_val = min_val = danh_sach[0]
11
12    for gia_tri in danh_sach[1:]:
13        if gia_tri > max_val:
14            max_val = gia_tri
15        if gia_tri < min_val:
16            min_val = gia_tri
17
18    return max_val, min_val
19
20 def tim_kiem_tuyen_tinh(danh_sach, gia_tri):
21     """
22         Tim kiem tuyen tinh
23         Input: danh_sach (list), gia_tri - gia tri can tim
24         Output: int - vi tri (index) hoac -1 neu khong tim thay
25         """
26     for i in range(len(danh_sach)):
27         if danh_sach[i] == gia_tri:
28             return i
29     return -1
30
31 def sap_xep_noi_bot(danh_sach):
32     """
33         Sap xep noi bot (bubble sort)
34         Input: danh_sach (list)
35         Output: list - danh sach da sap xep
36         """
37         # Tao ban sao de khong thay doi danh sach goc
38 arr = danh_sach.copy()
39 n = len(arr)
40
41 for i in range(n):
42     # Co co hoan vi nao khong
43     co_hoan_vi = False
44
45     for j in range(0, n - i - 1):
46         if arr[j] > arr[j + 1]:
47             # Hoan vi
48             arr[j], arr[j + 1] = arr[j + 1], arr[j]

```

```
19         co_hoan_vi = True
20
21     # Neu khong co hoan vi nao, da sap xep xong
22     if not co_hoan_vi:
23         break
24
25     return arr
26
27
28 def sap_xep_chon(danh_sach):
29     """
30     Sap xep chon (selection sort)
31     Input: danh_sach (list)
32     Output: list - danh sach da sap xep
33     """
34     arr = danh_sach.copy()
35     n = len(arr)
36
37     for i in range(n):
38         # Tim vi tri phan tu nho nhat trong phan con lai
39         min_idx = i
40         for j in range(i + 1, n):
41             if arr[j] < arr[min_idx]:
42                 min_idx = j
43
44         # Hoan vi phan tu nho nhat ve dau
45         arr[i], arr[min_idx] = arr[min_idx], arr[i]
46
47     return arr
48
49
50 def sap_xep_chen(danh_sach):
51     """
52     Sap xep chen (insertion sort)
53     Input: danh_sach (list)
54     Output: list - danh sach da sap xep
55     """
56     arr = danh_sach.copy()
57
58     for i in range(1, len(arr)):
59         key = arr[i]
60         j = i - 1
61
62         # Dich chuyen cac phan tu > key ve phia sau
63         while j >= 0 and arr[j] > key:
64             arr[j + 1] = arr[j]
65             j -= 1
66
67         arr[j + 1] = key
68
69     return arr
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
```

```
99 # Test cac ham
100 print("==== BAI 61: TIM KIEM VA SAP XEP ====")
101
102 test_list = [64, 34, 25, 12, 22, 11, 90]
103 print(f"Danh sach goc: {test_list}")
104
105 # Tim max min
106 max_val, min_val = tim_max_min(test_list)
107 print(f"Max: {max_val}, Min: {min_val}")
108
109 # Tim kiem
110 gia_tri_tim = 25
111 vi_tri = tim_kiem_tuyen_tinh(test_list, gia_tri_tim)
112 if vi_tri != -1:
113     print(f"Tim thay {gia_tri_tim} tai vi tri {vi_tri}")
114 else:
115     print(f"Khong tim thay {gia_tri_tim}")
116
117 # Sap xep
118 print(f"Bubble sort: {sap_xep_noi_bot(test_list)}")
119 print(f"Selection sort: {sap_xep_chon(test_list)}")
120 print(f"Insertion sort: {sap_xep_chen(test_list)}")
121
122 # So sanh hieu suat (don gian)
123 import time
124
125 def do_hieu_suat_sap_xep(danh_sach, ham_sap_xep, ten_ham):
126     """Do thoi gian thuc thi ham sap xep"""
127     start = time.time()
128     ket_qua = ham_sap_xep(danh_sach)
129     end = time.time()
130     print(f"{ten_ham}: {(end - start) * 1000:.2f}ms")
131     return ket_qua
132
133 # Test voi danh sach lon hon
134 import random
135 big_list = [random.randint(1, 1000) for _ in range(100)]
136
137 print(f"\nTest hieu suat voi {len(big_list)} phan tu:")
138 do_hieu_suat_sap_xep(big_list, sap_xep_noi_bot, "Bubble
139 Sort")
140 do_hieu_suat_sap_xep(big_list, sap_xep_chon, "Selection
141 Sort")
142 do_hieu_suat_sap_xep(big_list, sap_xep_chen, "Insertion
143 Sort")
144 do_hieu_suat_sap_xep(big_list, sorted, "Python Built-in")
```

## Bài Tập

### Bài 62: Hàm đệ quy

Viết các hàm đệ quy:

- `fibonacci_de_quy(n)`: Tính số Fibonacci thứ n
- `tinh_luy_thua(co_so, mu)`: Tính lũy thừa bằng đệ quy
- `dao_nguoc_chuoi_de_quy(s)`: Đảo ngược chuỗi bằng đệ quy
- `gcd_de_quy(a, b)`: Tính UCLN bằng thuật toán Euclid đệ quy

Ví dụ:

`fibonacci_de_quy(6) → 8`  
`tinh_luy_thua(2, 3) → 8`

## Hướng Dẫn

### Hướng dẫn:

1. Xác định điều kiện dừng (base case)
2. Xác định cách chia nhỏ bài toán (recursive case)
3. Chú ý tránh đệ quy vô hạn
4. Đệ quy có thể chậm với số lớn, cân nhắc sử dụng memoization

## Code Mẫu

```

1 def fibonacci_de_quy(n):
2     """
3         Tính số Fibonacci thu n bằng đệ quy
4         Input: n (int) - vị trí trong dãy Fibonacci
5         Output: int - số Fibonacci
6     """
7     # Điều kiện dừng
8     if n <= 1:
9         return n
10
11    # Đệ quy
12    return fibonacci_de_quy(n - 1) + fibonacci_de_quy(n - 2)
13
14 def fibonacci_memo(n, memo={}):
15     """
16         Fibonacci với memoization để tối ưu hiệu suất
17     """
18     if n in memo:
19         return memo[n]
20

```

```
21     if n <= 1:
22         return n
23
24     memo[n] = fibonacci_memo(n - 1, memo) + fibonacci_memo(n
25         - 2, memo)
26     return memo[n]
27
28 def tinh_luy_thua(co_so, mu):
29     """
30     Tinh luy thua bang de quy
31     Input: co_so, mu (int)
32     Output: int - ket qua co_so^mu
33     """
34
35     # Dieu kien dung
36     if mu == 0:
37         return 1
38     if mu == 1:
39         return co_so
40
41     # De quy
42     if mu % 2 == 0:
43         # Mu chan: a^n = (a^(n/2))^2
44         half = tinh_luy_thua(co_so, mu // 2)
45         return half * half
46     else:
47         # Mu le: a^n = a * a^(n-1)
48         return co_so * tinh_luy_thua(co_so, mu - 1)
49
50 def dao_nguoc_chuoi_de_quy(s):
51     """
52     Dao nguoc chuoi bang de quy
53     Input: s (str) - chuoi can dao nguoc
54     Output: str - chuoi da dao nguoc
55     """
56
57     # Dieu kien dung
58     if len(s) <= 1:
59         return s
60
61     # De quy: ky tu cuoi + dao nguoc phan con lai
62     return s[-1] + dao_nguoc_chuoi_de_quy(s[:-1])
63
64 def gcd_de_quy(a, b):
65     """
66     Tinh UCLN bang thuat toan Euclid de quy
67     Input: a, b (int) - hai so nguyen duong
68     Output: int - UCLN cua a va b
69     """
70
71     # Dieu kien dung
72     if b == 0:
73         return a
```

```
70
71     # De quy
72     return gcd_de_quy(b, a % b)
73
74 def tower_of_hanoi(n, source, destination, auxiliary):
75     """
76         Giai bài toán Tháp Hanoi bằng đệ quy
77         Input: n - số đĩa, source/destination/auxiliary - tên
78             các cột
79     """
80     if n == 1:
81         print(f"Chuyển đĩa 1 từ {source} sang {destination}")
82         return
83
84     # Chuyển n-1 đĩa từ source sang auxiliary
85     tower_of_hanoi(n - 1, source, auxiliary, destination)
86
87     # Chuyển đĩa lớn nhất từ source sang destination
88     print(f"Chuyển đĩa {n} từ {source} sang {destination}")
89
90     # Chuyển n-1 đĩa từ auxiliary sang destination
91     tower_of_hanoi(n - 1, auxiliary, destination, source)
92
93 # Test cac ham
94 print("== BAI 62: DE QUY ==")
95
96 # Test Fibonacci
97 print("Fibonacci (de quy don gian):")
98 for i in range(8):
99     print(f"F({i}) = {fibonacci_de_quy(i)}")
100
101 print("\nFibonacci (co memoization):")
102 for i in range(10):
103     print(f"F({i}) = {fibonacci_memo(i)}")
104
105 # Test luy thua
106 print(f"\n2^3 = {tinh_luy_thua(2, 3)}")
107 print(f"3^4 = {tinh_luy_thua(3, 4)}")
108 print(f"5^0 = {tinh_luy_thua(5, 0)}")
109
110 # Test dao nguoc chuoi
111 text = "Python"
112 print(f"\nĐảo ngược '{text}':\n'{dao_nguoc_chuoi_de_quy(text)}'")
113
114 # Test UCLN
115 print(f"\nUCLN(48, 18) = {gcd_de_quy(48, 18)}")
116 print(f"UCLN(100, 75) = {gcd_de_quy(100, 75)}")
117
118 # Demo Tháp Hanoi
```

```

118 print(f"\nGiai bai toan Thap Hanoi voi 3 dia:")
119 tower_of_hanoi(3, "A", "C", "B")
120
121 # So sanh hieu suat Fibonacci
122 import time
123
124 def so_sanh_fibonacci(n):
125     """So sanh hieu suat giua de quy va memoization"""
126     print(f"\nTinh Fibonacci({n}):")
127
128     # De quy don gian (chi test voi n nho)
129     if n <= 30:
130         start = time.time()
131         result1 = fibonacci_de_quy(n)
132         time1 = time.time() - start
133         print(f"De quy don gian: {result1} ({time1:.4f}s)")
134
135     # Memoization
136     start = time.time()
137     result2 = fibonacci_memo(n)
138     time2 = time.time() - start
139     print(f"Memoization: {result2} ({time2:.4f}s)")
140
141 so_sanh_fibonacci(20)
142 so_sanh_fibonacci(35) # Chi chay memoization

```

## Bài Tập

### Bài 63: Hàm xử lý danh sách nâng cao

Viết các hàm:

- loc\_so\_chan(danh\_sach): Lọc ra các số chẵn
- bien\_doi\_danh\_sach(danh\_sach, ham\_bien\_doi): Áp dụng hàm lên từng phần tử
- gom\_nhom\_theo\_dieu\_kien(danh\_sach, ham\_dieu\_kien): Nhóm theo điều kiện
- tinh\_thong\_ke(danh\_sach): Tính mean, median, mode

Ví dụ:

`loc_so_chan([1,2,3,4,5]) → [2,4]`  
`bien_doi_danh_sach([1,2,3], lambda x: x*2) → [2,4,6]`

## Hướng Dẫn

### Hướng dẫn:

1. Sử dụng list comprehension cho code ngắn gọn
2. Tìm hiểu về lambda functions
3. Sử dụng filter(), map(), reduce() khi phù hợp
4. Tính median: sắp xếp rồi lấy phần tử giữa

## Code Mẫu

```

1  from collections import Counter
2  import statistics
3
4  def loc_so_chan(danh_sach):
5      """
6          Loc ra cac so chan
7          Input: danh_sach (list)
8          Output: list - cac so chan
9      """
10     return [x for x in danh_sach if x % 2 == 0]
11
12 def loc_theo_dieu_kien(danh_sach, dieu_kien):
13     """
14         Loc danh sach theo dieu kien tuy chinh
15         Input: danh_sach (list), dieu_kien (function)
16         Output: list - cac phan tu thoa man dieu kien
17     """
18     return [x for x in danh_sach if dieu_kien(x)]
19
20 def bien_doi_danh_sach(danh_sach, ham_bien_doi):
21     """
22         Ap dung ham bien doi len tung phan tu
23         Input: danh_sach (list), ham_bien_doi (function)
24         Output: list - danh sach da bien doi
25     """
26     return [ham_bien_doi(x) for x in danh_sach]
27
28 def gom_nhom_theo_dieu_kien(danh_sach, ham_dieu_kien):
29     """
30         Gom nhom theo dieu kien
31         Input: danh_sach (list), ham_dieu_kien (function) -> bool
32         Output: tuple - (nhom_true, nhom_false)
33     """
34     nhom_true = []
35     nhom_false = []
36
37     for item in danh_sach:

```

```
38     if ham_dieu_kien(item):
39         nhom_true.append(item)
40     else:
41         nhom_false.append(item)
42
43     return nhom_true, nhom_false
44
45 def tinh_thong_ke(danh_sach):
46     """
47     Tinh cac thong ke co ban
48     Input: danh_sach (list) - danh sach so
49     Output: dict - cac chi so thong ke
50     """
51     if not danh_sach:
52         return None
53
54     # Sap xep de tinh median
55     sorted_list = sorted(danh_sach)
56     n = len(sorted_list)
57
58     # Mean (trung binh)
59     mean = sum(danh_sach) / n
60
61     # Median (trung vi)
62     if n % 2 == 0:
63         median = (sorted_list[n//2 - 1] + sorted_list[n//2])
64             / 2
65     else:
66         median = sorted_list[n//2]
67
68     # Mode (gia tri xuat hien nhieu nhat)
69     counter = Counter(danh_sach)
70     max_count = max(counter.values())
71     mode = [k for k, v in counter.items() if v == max_count]
72
73     # Range (khoang gia tri)
74     range_val = max(danh_sach) - min(danh_sach)
75
76     # Variance va Standard Deviation
77     variance = sum((x - mean) ** 2 for x in danh_sach) / n
78     std_dev = variance ** 0.5
79
80     return {
81         'count': n,
82         'sum': sum(danh_sach),
83         'mean': mean,
84         'median': median,
85         'mode': mode,
86         'min': min(danh_sach),
87         'max': max(danh_sach),
```

```
87         'range': range_val,
88         'variance': variance,
89         'std_dev': std_dev
90     }
91
92 def tim_phan_tu_doc_nhat(danh_sach):
93     """Tim phan tu xuat hien duy nhat 1 lan"""
94     counter = Counter(danh_sach)
95     return [k for k, v in counter.items() if v == 1]
96
97 def hop_hai_danh_sach_sap_xep(list1, list2):
98     """
99     Hop hai danh sach da sap xep thanh mot danh sach sap xep
100    """
101    result = []
102    i = j = 0
103
104    while i < len(list1) and j < len(list2):
105        if list1[i] <= list2[j]:
106            result.append(list1[i])
107            i += 1
108        else:
109            result.append(list2[j])
110            j += 1
111
112    # Them cac phan tu con lai
113    result.extend(list1[i:])
114    result.extend(list2[j:])
115
116    return result
117
118 # Test cac ham
119 print("==== BAI 63: XU LY DANH SACH NANG CAO ===")
120
121 test_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
122 print(f"Danh sach goc: {test_list}")
123
124 # Loc so chan
125 so_chan = loc_so_chan(test_list)
126 print(f"So chan: {so_chan}")
127
128 # Loc so nguyen to
129 so_nguyen_to = loc_theo_dieu_kien(test_list, lambda x:
130     kiem_tra_nguyen_to(x))
131 print(f"So nguyen to: {so_nguyen_to}")
132
133 # Bien doi: binh phuong
134 binh_phuong = bien_doi_danh_sach(test_list, lambda x: x**2)
135 print(f"Binh phuong: {binh_phuong}")
```

```

136 # Gom nhom chan le
137 nhom_chan, nhom_le = gom_nhom_theo_dieu_kien(test_list,
138     lambda x: x % 2 == 0)
139 print(f"Nhóm chẵn: {nhom_chan}")
140 print(f"Nhóm lẻ: {nhom_le}")

141 # Thống kê
142 data = [1, 2, 2, 3, 4, 4, 4, 5, 6, 7, 8, 9]
143 stats = tinh_thong_ke(data)
144 print(f"\nThống kê cho {data}:")
145 for key, value in stats.items():
146     print(f"  {key}: {value}")

147 # Tìm phần tử độc nhất
148 doc_nhat = tim_phan_tu_doc_nhat([1, 2, 2, 3, 3, 4, 5])
149 print(f"\nPhần tử độc nhất: {doc_nhat}")

150 # Hợp hai danh sách
151 list1 = [1, 3, 5, 7]
152 list2 = [2, 4, 6, 8, 10]
153 merged = hop_hai_danh_sach.sap_xep(list1, list2)
154 print(f"Hợp {list1} và {list2}: {merged}")

155 # Sử dụng built-in functions
156 print(f"\nSử dụng filter(): {list(filter(lambda x: x > 5,
157     test_list))}")
158 print(f"Sử dụng map(): {list(map(lambda x: x * 3, [1, 2,
159     3]))}")

160 # Sử dụng module statistics
161 try:
162     print(f"Statistics module - mean:
163         {statistics.mean(data)}")
164     print(f"Statistics module - median:
165         {statistics.median(data)}")
166     print(f"Statistics module - mode:
167         {statistics.mode(data)}")
168 except statistics.StatisticsError as e:
169     print(f"Statistics error: {e}")

```

## Bài Tập

### Bài 64: Hàm mô phỏng trò chơi

Viết hàm mô phỏng các trò chơi:

- tung\_xu(so\_lan): Mô phỏng tung xu
- tung\_xuc\_xac(so\_mat, so\_lan): Mô phỏng tung xúc xác
- doan\_so(min\_val, max\_val): Game đoán số

- `bai_cao_thap()`: Game bài cao thấp

Ví dụ:

```
tung_xu(100) → {"sap": 52, "ngua": 48}
tung_xuc_xac(6, 10) → [3, 1, 6, 2, 5, 1, 4, 6, 3, 2]
```

## Hướng Dẫn

### Hướng dẫn:

1. Sử dụng module random để tạo số ngẫu nhiên
2. Tạo giao diện đơn giản với input/output
3. Lưu trữ kết quả và thống kê
4. Xử lý input người dùng và validate

## Code Mẫu

```

1 import random
2 import time
3
4 def tung_xu(so_lan=1):
5     """
6         Mo phong tung xu
7         Input: so_lan (int) - so lan tung
8         Output: dict - ket qua thong ke
9     """
10    ket_qua = []
11
12    for _ in range(so_lan):
13        # 0 = sap, 1 = ngua
14        result = random.choice([0, 1])
15        ket_qua.append("sap" if result == 0 else "ngua")
16
17    # Thong ke
18    sap = ket_qua.count("sap")
19    ngua = ket_qua.count("ngua")
20
21    return {
22        "ket_qua": ket_qua,
23        "sap": sap,
24        "ngua": ngua,
25        "tie_le_sap": sap / so_lan * 100,
26        "tie_le_ngua": ngua / so_lan * 100
27    }
28
29 def tung_xuc_xac(so_mat=6, so_lan=1):
30     """

```

```
31     Mo phong tung xuc xac
32     Input: so_mat (int) - so mat xuc xac, so_lan (int) - so
33         lan tung
34     Output: list - cac ket qua
35     """
36     ket_qua = []
37     thong_ke = {i: 0 for i in range(1, so_mat + 1)}
38
39     for _ in range(so_lan):
40         roll = random.randint(1, so_mat)
41         ket_qua.append(roll)
42         thong_ke[roll] += 1
43
44     return {
45         "ket_qua": ket_qua,
46         "thong_ke": thong_ke,
47         "tong": sum(ket_qua),
48         "trung_binh": sum(ket_qua) / so_lan
49     }
50
51 def doan_so(min_val=1, max_val=100):
52     """
53     Game doan so
54     Input: min_val, max_val - khoang gia tri
55     Output: dict - ket qua game
56     """
57     so_can_doan = random.randint(min_val, max_val)
58     so_lan_doan = 0
59     lich_su_doan = []
60
61     print(f"==== GAME DOAN SO ====")
62     print(f"Toi da nghi ra 1 so tu {min_val} den {max_val}")
63     print("Ban hay doan xem do la so nao!")
64
65     while True:
66         try:
67             doan = int(input("Nhap so doan cua ban: "))
68             so_lan_doan += 1
69             lich_su_doan.append(doan)
70
71             if doan == so_can_doan:
72                 print(f"Chinh xac! Ban da doan dung sau
73                     {so_lan_doan} lan.")
74                 break
75             elif doan < so_can_doan:
76                 print("So ban doan nho hon ket qua.")
77             else:
78                 print("So ban doan lon hon ket qua.")
```

```
79         print("Vui long nhap mot so hop le!")
80
81     return {
82         "so_dung": so_can_doan,
83         "so_lan_doan": so_lan_doan,
84         "lich_su": lich_su_doan,
85         "thanhCong": True
86     }
87
88 def bai_cao_thap():
89     """
90     Game bai cao thap
91     Nguoi choi doan la bai tiep theo cao hon hay thap hon
92     bai hien tai
93     """
94     bai = list(range(1, 14)) * 4    # A=1, J=11, Q=12, K=13
95     random.shuffle(bai)
96
97     diem = 0
98     vi_tri = 0
99
100    print("==== GAME BAI CAO THAP ===")
101    print("Doan bai tiep theo cao hon (C) hay thap hon (T)
102        bai hien tai")
103    print("A=1, J=11, Q=12, K=13")
104
105    while vi_tri < len(bai) - 1:
106        bai_hien_tai = bai[vi_tri]
107        bai_tiep_theo = bai[vi_tri + 1]
108
109        print(f"\nBai hien tai: {bai_hien_tai}")
110        print(f"Diem hien tai: {diem}")
111
112        doan = input("Bai tiep theo Cao hon (C) hay Thap hon
113                    (T)? ").upper()
114
115        if doan not in ['C', 'T']:
116            print("Vui long nhap C hoac T!")
117            continue
118
119            print(f"Bai tiep theo: {bai_tiep_theo}")
120
121            dung = False
122            if doan == 'C' and bai_tiep_theo > bai_hien_tai:
123                dung = True
124            elif doan == 'T' and bai_tiep_theo < bai_hien_tai:
125                dung = True
126            elif bai_tiep_theo == bai_hien_tai:
127                print("Bang nhau! Tiep tuc...")
128                vi_tri += 1
```

```

126         continue
127
128     if dung:
129         diem += 1
130         print("Dung roi! +1 diem")
131     else:
132         diem -= 1
133         print("Sai roi! -1 diem")
134
135     vi_tri += 1
136
137     # Hỏi có muốn tiếp tục không
138     if vi_tri < len(bai) - 1:
139         tiep_tuc = input("Tiếp tục? (Y/N): ").upper()
140         if tiep_tuc == 'N':
141             break
142
143     print(f"\nGame kết thúc! Điểm cuối cùng: {diem}")
144     return {"diem_cuoi": diem, "so_bai_choi": vi_tri}
145
146 def xo_so_mini():
147     """
148     Game xo so mini - chọn 6 số từ 1-49
149     """
150     print("==== XO SO MINI ====")
151     print("Chọn 6 số từ 1 đến 49")
152
153     # Người chơi chọn số
154     so_chon = []
155     for i in range(6):
156         while True:
157             try:
158                 so = int(input(f"Chọn số thứ {i+1}: "))
159                 if 1 <= so <= 49 and so not in so_chon:
160                     so_chon.append(so)
161                     break
162                 else:
163                     print("Số phải từ 1-49 và chưa được
164                     chọn!")
165             except ValueError:
166                 print("Vui lòng nhập số hợp lệ!")
167
168     # Quay số
169     print("\nĐang quay số...")
170     time.sleep(2)
171
172     so_quay = random.sample(range(1, 50), 6)
173     so_quay.sort()
174
175     print(f"So ban chon: {sorted(so_chon)}")

```

```

175     print(f"So quay duoc: {so_quay}")
176
177     # Kiem tra trung
178     so_trung = set(so_chon) & set(so_quay)
179
180     print(f"Ban trung {len(so_trung)} so:
181         {sorted(list(so_trung))}")
182
183     # Tinh thuong
184     thuong = {
185         6: "Giai nhat - 1 ty dong!",
186         5: "Giai nhi - 10 trieu dong!",
187         4: "Giai ba - 1 trieu dong!",
188         3: "Giai khuyem khuyen - 100k dong!",
189         2: "An ui - 10k dong",
190         1: "Chuc ban may man lan sau!",
191         0: "Chuc ban may man lan sau!"
192     }
193
194     print(thuong[len(so_trung)])
195
196     return {
197         "so_chon": so_chon,
198         "so_quay": so_quay,
199         "so_trung": len(so_trung),
200         "thuong": thuong[len(so_trung)]
201     }
202
203     # Test cac ham (khong goi game tuong tac)
204     print("== BAI 64: MO PHONG TRO CHOI ==")
205
206     # Test tung xu
207     print("Test tung xu 20 lan:")
208     ket_qua_xu = tung_xu(20)
209     print(f"Sap: {ket_qua_xu['sap']} lan
210         ({ket_qua_xu['ti_le_sap']:.1f}%)")
211     print(f"Ngua: {ket_qua_xu['ngua']} lan
212         ({ket_qua_xu['ti_le_ngua']:.1f}%)")
213
214     # Test tung xuc xac
215     print(f"\nTest tung xuc xac 6 mat, 10 lan:")
216     ket_qua_xuc_xac = tung_xuc_xac(6, 10)
217     print(f"Ket qua: {ket_qua_xuc_xac['ket_qua']}")
218     print(f"Tong: {ket_qua_xuc_xac['tong']}")
219     print(f"Trung binh: {ket_qua_xuc_xac['trung_binh']:.2f}")
220     print("Thong ke:")
221     for mat, lan in ket_qua_xuc_xac['thong_ke'].items():
222         print(f"    Mat {mat}: {lan} lan")
223
224     # Ham auto-play cho game doan so (khong can input)

```

```

222 def auto_doan_so(min_val=1, max_val=100):
223     """Version tu dong cua game doan so"""
224     so_can_doan = random.randint(min_val, max_val)
225     so_lan_doan = 0
226     low, high = min_val, max_val
227
228     while True:
229         # Chien luoc binary search
230         doan = (low + high) // 2
231         so_lan_doan += 1
232
233         if doan == so_can_doan:
234             break
235         elif doan < so_can_doan:
236             low = doan + 1
237         else:
238             high = doan - 1
239
240     return {
241         "so_dung": so_can_doan,
242         "so_lan_doan": so_lan_doan,
243         "chien_luoc": "Binary Search"
244     }
245
246 print(f"\nAuto-play doan so:")
247 auto_result = auto_doan_so()
248 print(f"So dung: {auto_result['so_dung']}")
249 print(f"Doan dung sau: {auto_result['so_lan_doan']} lan")
250
251 # Uncomment de choi cac game tuong tac:
252 # doan_so()
253 # bai_cao_thap()
254 # xo_so_mini()

```

## Bài Tập

### Bài 65: Hàm quản lý dữ liệu học sinh

Tạo hệ thống quản lý học sinh với các hàm:

- them\_hoc\_sinh(danh\_sach, ten, tuoi, diem): Thêm học sinh
- tim\_hoc\_sinh(danh\_sach, ten): Tìm học sinh theo tên
- xep\_loai\_hoc\_sinh(diem): Xếp loại theo điểm
- thong\_ke\_lop(danh\_sach): Thống kê toàn lớp
- sap\_xep\_theo\_diem(danh\_sach): Sắp xếp theo điểm

#### Ví dụ:

Quản lý danh sách học sinh với thông tin đầy đủ và các thao tác CRUD cơ bản.

## Hướng Dẫn

### Hướng dẫn:

1. Sử dụng dictionary để lưu thông tin học sinh
2. Tạo các hàm helper để xử lý dữ liệu
3. Validate input trước khi xử lý
4. Tạo menu đơn giản cho người dùng

## Code Mẫu

```
1 import json
2 from datetime import datetime
3
4 def tao_hoc_sinh(ten, tuoi, diem_toan, diem_ly, diem_hoa):
5     """
6         Tao thong tin hoc sinh
7         Input: ten (str), tuoi (int), diem_toan, diem_ly,
8             diem_hoa (float)
9         Output: dict - thong tin hoc sinh
10    """
11    if not ten.strip():
12        raise ValueError("Ten khong duoc rong")
13    if tuoi < 0 or tuoi > 100:
14        raise ValueError("Tuoi phai tu 0-100")
15    if not all(0 <= diem <= 10 for diem in [diem_toan,
16        diem_ly, diem_hoa]):
17        raise ValueError("Diem phai tu 0-10")
18
19    diem_tb = (diem_toan + diem_ly + diem_hoa) / 3
20
21    return {
22        "id": int(datetime.now().timestamp() * 1000) %
23            100000, # ID don gian
24        "ten": ten.strip().title(),
25        "tuoi": tuoi,
26        "diem": {
27            "toan": diem_toan,
28            "ly": diem_ly,
29            "hoa": diem_hoa,
30            "trung_binh": round(diem_tb, 2)
31        },
32        "xep_loai": xep_loai_hoc_sinh(diem_tb),
33        "ngay_tao": datetime.now().strftime("%d/%m/%Y
34                                     %H:%M:%S")
35    }
36
37 def them_hoc_sinh(danh_sach, ten, tuoi, diem_toan, diem_ly,
```

```
34     diem_hoa):
35         """
36         Them hoc sinh vao danh sach
37         """
38     try:
39         hoc_sinh = tao_hoc_sinh(ten, tuoi, diem_toan,
40                                  diem_ly, diem_hoa)
41         danh_sach.append(hoc_sinh)
42         return True, f"Da them hoc sinh {ten}"
43     except ValueError as e:
44         return False, str(e)
45
46 def tim_hoc_sinh(danh_sach, ten=None, hoc_sinh_id=None):
47     """
48     Tim hoc sinh theo ten hoac ID
49     """
50     ket_qua = []
51
52     for hs in danh_sach:
53         if ten and ten.lower() in hs["ten"].lower():
54             ket_qua.append(hs)
55         elif hoc_sinh_id and hs["id"] == hoc_sinh_id:
56             ket_qua.append(hs)
57
58     return ket_qua
59
60 def xep_loai_hoc_sinh(diem_tb):
61     """
62     Xep loai hoc sinh theo diem trung binh
63     """
64     if diem_tb >= 9.0:
65         return "Xuat sac"
66     elif diem_tb >= 8.0:
67         return "Gioi"
68     elif diem_tb >= 6.5:
69         return "Kha"
70     elif diem_tb >= 5.0:
71         return "Trung binh"
72     else:
73         return "Yeu"
74
75 def cap_nhat_hoc_sinh(danh_sach, hoc_sinh_id, **kwargs):
76     """
77     Cap nhat thong tin hoc sinh
78     """
79     for i, hs in enumerate(danh_sach):
80         if hs["id"] == hoc_sinh_id:
81             # Cap nhat cac truong cho phep
```

```
                                kwargs["ten"].strip().title()
82    if "tuoi" in kwargs:
83        danh_sach[i]["tuoi"] = kwargs["tuoi"]
84
85    # Cap nhat diem
86    diem_update = False
87    for mon in ["toan", "ly", "hoa"]:
88        if mon in kwargs:
89            danh_sach[i]["diem"][mon] = kwargs[mon]
90            diem_update = True
91
92    # Tinh lai diem trung binh neu co cap nhat diem
93    if diem_update:
94        diem = danh_sach[i]["diem"]
95        diem_tb = (diem["toan"] + diem["ly"] +
96                    diem["hoa"]) / 3
97        danh_sach[i]["diem"]["trung_binh"] =
98            round(diem_tb, 2)
99        danh_sach[i]["xep_loai"] =
100            xep_loai_hoc_sinh(diem_tb)
101
102    return True, "Cap nhat thanh cong"
103
104
105 def xoa_hoc_sinh(danh_sach, hoc_sinh_id):
106     """
107     Xoa hoc sinh khoi danh sach
108     """
109     for i, hs in enumerate(danh_sach):
110         if hs["id"] == hoc_sinh_id:
111             ten = hs["ten"]
112             del danh_sach[i]
113             return True, f"Da xoa hoc sinh {ten}"
114
115     return False, "Khong tim thay hoc sinh"
116
117
118 def sap_xep_theo_diem(danh_sach, giam_dan=True):
119     """
120     Sap xep danh sach theo diem trung binh
121     """
122     return sorted(danh_sach,
123                   key=lambda x: x["diem"]["trung_binh"],
124                   reverse=giam_dan)
125
126
127 def thong_ke_lop(danh_sach):
128     """
129     Thong ke toan lop
130     """
131     if not danh_sach:
```

```
128     return {"thong_bao": "Danh sach rong"}
129
130     # Tinh toan cac chi so
131     tong_hs = len(danh_sach)
132     diem_tb_lop = sum(hs["diem"]["trung_binh"] for hs in
133                     danh_sach) / tong_hs
134
135     # Thong ke theo xep loai
136     xep_loai_count = {}
137     for hs in danh_sach:
138         xep_loai = hs["xep_loai"]
139         xep_loai_count[xep_loai] =
140             xep_loai_count.get(xep_loai, 0) + 1
141
142     # Tim hoc sinh co diem cao nhat/thap nhat
143     ds_sap_xep = sap_xep_theo_diem(danh_sach)
144     hs_gioi_nhat = ds_sap_xep[0]
145     hs_yeu_nhat = ds_sap_xep[-1]
146
147     # Thong ke theo mon
148     mon_tb = {}
149     for mon in ["toan", "ly", "hoa"]:
150         mon_tb[mon] = sum(hs["diem"][mon] for hs in
151                           danh_sach) / tong_hs
152
153     return {
154         "tong_hoc_sinh": tong_hs,
155         "diem_tb_lop": round(diem_tb_lop, 2),
156         "xep_loai": xep_loai_count,
157         "hoc_sinh_gioi_nhat": {
158             "ten": hs_gioi_nhat["ten"],
159             "diem": hs_gioi_nhat["diem"]["trung_binh"]
160         },
161         "hoc_sinh_yeu_nhat": {
162             "ten": hs_yeu_nhat["ten"],
163             "diem": hs_yeu_nhat["diem"]["trung_binh"]
164         },
165         "diem_tb_theo_mon": {mon: round(diem, 2) for mon,
166                               diem in mon_tb.items()}
167     }
168
169 def xuat_danh_sach(danh_sach, file_path="hoc_sinh.json"):
170     """
171     Xuat danh sach ra file JSON
172     """
173
174     try:
175         with open(file_path, 'w', encoding='utf-8') as f:
176             json.dump(danh_sach, f, ensure_ascii=False,
177                       indent=2)
178         return True, f"Da xuat ra file {file_path}"
179
```

```
173     except Exception as e:
174         return False, f"Loi xuat file: {e}"
175
176 def nhap_danh_sach(file_path="hoc_sinh.json"):
177     """
178     Nhập danh sách từ file JSON
179     """
180     try:
181         with open(file_path, 'r', encoding='utf-8') as f:
182             danh_sach = json.load(f)
183             return danh_sach, f"Da nhap tu file {file_path}"
184     except FileNotFoundError:
185         return [], "File khong ton tai"
186     except Exception as e:
187         return [], f"Loi doc file: {e}"
188
189 def in_hoc_sinh(hoc_sinh):
190     """
191     In thông tin 1 học sinh
192     """
193     print(f"ID: {hoc_sinh['id']} | Tên: {hoc_sinh['ten']} |"
194           f"Tuổi: {hoc_sinh['tuoi']}")
195     print(f"Điểm - Toán: {hoc_sinh['diem']['toan']}, Lý:"
196           f"{hoc_sinh['diem']['ly']}, Hoá:"
197           f"{hoc_sinh['diem']['hoa']}")
198     print(f"Điểm TB: {hoc_sinh['diem']['trung_binh']} | Xếp"
199           f"loại: {hoc_sinh['xep_loai']}")
200     print(f"Ngày tạo: {hoc_sinh['ngay_tao']}")
201     print("-" * 50)
202
203 def in_danh_sach(danh_sach):
204     """
205     In toàn bộ danh sách
206     """
207     if not danh_sach:
208         print("Danh sách rỗng!")
209         return
210
211     print(f"DANH SÁCH HỌC SINH ({len(danh_sach)} học sinh)")
212     print("=" * 50)
213     for hs in danh_sach:
214         in_hoc_sinh(hs)
215
216 # Test hệ thống quản lý học sinh
217 print("==== BÀI 65: QUẢN LÝ HỌC SINH ====")
218
219 # Tao danh sach hoc sinh mau
220 lop_10a = []
221
222 # Them hoc sinh
```

```
219 print("Them hoc sinh vao lop:")
220 hoc_sinh_mau = [
221     ("Nguyen Van An", 16, 8.5, 7.5, 9.0),
222     ("Tran Thi Binh", 15, 9.0, 8.5, 8.0),
223     ("Le Minh Duc", 16, 7.0, 6.5, 7.5),
224     ("Pham Thu Huong", 15, 9.5, 9.0, 9.5),
225     ("Hoang Van Dat", 16, 6.0, 5.5, 6.5)
226 ]
227
228 for ten, tuoi, toan, ly, hoa in hoc_sinh_mau:
229     success, msg = them_hoc_sinh(lop_10a, ten, tuoi, toan,
230                                   ly, hoa)
231     print(f" {msg}")
232
233 print(f"\nDanh sach hien tai:")
234 in_danh_sach(lop_10a)
235
236 # Thong ke lop
237 print("THONG KE LOP:")
238 stats = thong_ke_lop(lop_10a)
239 print(f"Tong hoc sinh: {stats['tong_hoc_sinh']} ")
240 print(f"Diem TB lop: {stats['diem_tb_lop']} ")
241 print(f"HS gioi nhat: {stats['hoc_sinh_gioi_nhat']['ten']} "
242       f"({stats['hoc_sinh_gioi_nhat']['diem']} )")
243 print(f"HS can co gang: {stats['hoc_sinh_yeu_nhat']['ten']} "
244       f"({stats['hoc_sinh_yeu_nhat']['diem']} )")
245 print("Xep loai:")
246 for loai, so_luong in stats['xep_loai'].items():
247     print(f" {loai}: {so_luong} HS")
248 print("Diem TB theo mon:")
249 for mon, diem in stats['diem_tb_theo_mon'].items():
250     print(f" {mon.title()} : {diem}")
251
252 # Tim kiem
253 print(f"\nTim kiem hoc sinh co ten 'An':")
254 ket_qua = tim_hoc_sinh(lop_10a, ten="An")
255 for hs in ket_qua:
256     print(f" {hs['ten']} - DTB: {hs['diem']['trung_binh']} ")
257
258 # Sap xep theo diem
259 print(f"\nDANH SACH SAP XEP THEO DIEM (Giam dan):")
260 ds_sap_xep = sap_xep_theo_diem(lop_10a)
261 for i, hs in enumerate(ds_sap_xep, 1):
262     print(f"{i}. {hs['ten']} - DTB: "
263           f"{hs['diem']['trung_binh']} ({hs['xep_loai']})")
264
265 # Cap nhat thong tin
266 if lop_10a:
267     hs_dau_tien = lop_10a[0]
268     print(f"\nCap nhat diem Toan cho {hs_dau_tien['ten']} tu
```

```
265     {hs_dau_tien['diem']['toan']} thanh 10:")
266 success, msg = cap_nhat_hoc_sinh(lop_10a,
267     hs_dau_tien['id'], toan=10.0)
268 print(f" {msg}")
269 print(f" Diem TB moi:
270     {lop_10a[0]['diem']['trung_binh']}")
271 print(f" Xep loai moi: {lop_10a[0]['xep_loai']}")
```

## 1.8 Tổng Kết Chương Hành

### Lý Thuyết

Kiến thức đã học trong chương này:

#### 1. Các loại hàm:

- Hàm built-in của Python
- Hàm từ thư viện (module)
- Hàm tự định nghĩa

#### 2. Thành phần của hàm:

- Định nghĩa hàm với `def`
- Tham số và đối số
- Docstring và `return`
- Phạm vi biến (scope)

#### 3. Kỹ thuật nâng cao:

- Dệ quy (recursion)
- Lambda functions
- `*args` và `**kwargs`
- Memoization

#### 4. Ứng dụng thực tế:

- Xử lý chuỗi và danh sách
- Tính toán toán học
- Mô phỏng trò chơi
- Quản lý dữ liệu

Lợi ích của việc sử dụng hàm:

- **Tái sử dụng:** Viết một lần, dùng nhiều lần
- **Modular:** Chia code thành các phần nhỏ, dễ quản lý
- **Debug dễ dàng:** Tách biệt logic, dễ tìm lỗi
- **Đọc hiểu tốt:** Code rõ ràng, có ý nghĩa
- **Testing:** Dễ dàng test từng phần riêng biệt

Best practices khi viết hàm:

- Tên hàm rõ ràng, mô tả chức năng
- Một hàm chỉ làm một việc (Single Responsibility)
- Viết docstring mô tả hàm
- Validate input trước khi xử lý
- Return giá trị phù hợp
- Tránh side effects không mong muốn

## CHÚC MỪNG!

Bạn đã hoàn thành chương về Hàm trong Python!  
*Hàm là nền tảng của lập trình modular. Hãy tiếp tục thực hành!*