Trên lớp

- Dùng hàm ẩn danh

lambda arg1, arg2: biểu thức

- VD: lambda a:a**2

def binhphuong(a): return a**2

In [ ]:
```
binhphuong = lambda a: a**2
binhphuong(5)
```

Out[ ]: 25

In [2]:
```
#Bài 24
def factorial(n):
  """ Hàm tính n giai thừa bằng đệ quy!"""
  if n < 0:
    return -1
  elif n == 0:
    return 1
  else:
    return n * factorial(n-1)
```

In [3]:
```
  print("5 giai thừa tính theo đệ quy: ", factorial(5))
  print("4 giai thừa tính theo đệ quy: ", factorial(4))
```

```
5 giai thừa tính theo đệ quy:  120
4 giai thừa tính theo đệ quy:  24
```

Bài tập Thực hành Python Bài số 4

Phần A

In [9]:
```
# Bài 1
def hello():
  """ Hàm in ra dòng chữ Hello, World!"""
  print("Hello, World!")

hello()
```

```
Hello, World!
```

In [10]:
```
#Bài 2
def add(a, b):
  """ Hàm tính tổng 2 số """
  return a+b

a, b = 1, 2
print("a + b = ", add(a, b))
```

```
a + b =  3
```

In [11]:
```python
# Bài 3
def greet(name = "Python"):
  """ Hàm có tham số mặc định name="Python" để in ra thông điệp chào """
  print(f"Xin chào {name}")

greet()
greet("Lan")
```

```
Xin chào Python
Xin chào Lan
```

Bài 4

- Positional argument (tham số vị trí): Khi gọi hàm, các positional argument được gán cho các tham số của hàm dựa trên vị trí của chúng. Thứ tự của các argument phải khớp với thứ tự của các tham số trong định nghĩa hàm, không cần sử dụng tên của tham số khi truyền positional argument.
- Keyword argument (tham số từ khóa):

Khi gọi hàm, các keyword argument được gán cho các tham số của hàm bằng cách sử dụng tên của tham số, theo sau là dấu bằng (=) và giá trị, không quan trọng thứ tự.

In [12]:
```python
# Bài 5
def describe_pet(animal_type, pet_name):
  """ Hiển thị thông tin thú cưng """
  print(f"Tôi có một con {animal_type}")
  print(f"Tên của nó là {pet_name}")

# Gọi hàm sử dụng positional argument
describe_pet("chó", "Nui")

# Gọi hàm sử keyword argument
describe_pet(pet_name= "Nui", animal_type= "chó")
```

```
Tôi có một con chó
Tên của nó là Nui
Tôi có một con chó
Tên của nó là Nui
```

In [13]:
```python
# Bài 6
def power(base, exp = 2):
  return base**exp

power(3, 2)
```

Out[13]: 9

In [14]:
```python
# Bài 7
def print_strings(*args, sep=' '):
  """ Định nghĩa một hàm với keyword-only argument *, sep để in nhiều chuỗi """
  print(*args, sep=sep)

# Ví dụ sử dụng
print_strings("Xin", "chào", "thế", "giới", sep="-")
print_strings("Một", "hai", "ba")
```

```
Xin-chào-thế-giới
Một hai ba
```

In [15]:
```python
# Bài 8
def nhan(a,b, /):
  return a*b

print(nhan(10, 2))
```

```
20
```

In [16]:
```python
# lỗi nếu truyền theo keyword argument
print(nhan(b=10, a=2)
```

```
  File "/tmp/ipython-input-2801192325.py", line 2
    print(nhan(b=10, a=2)
                         ^
SyntaxError: incomplete input
```

In [17]:
```python
# Bài 9: Viết một hàm sum_all(*args) tính tổng các số truyền vào.
def sum_all(*args):
  """ Tính tổng các số truyền vào sử dụng *args """
  total = 0
  for num in args:
    total += num
  return total

print("Tổng của 1, 2, 3 ,4, 5 là: ", sum_all(1, 2, 3, 4, 5))
print("Tổng của 1, 10, 13 là: ", sum_all(1, 10, 13))
```

```
Tổng của 1, 2, 3 ,4, 5 là:  15
Tổng của 1, 10, 13 là:  24
```

In [18]:
```python
# Bài 10: Viết một hàm print_info(**kwargs) in ra toàn bộ thông tin nhận được.

def print_info(**kwargs):
  """ In ra thông tin nhận được dưới dạng keywords arguments """
  for key, value in kwargs.items():
    print(f"{key}:{value}")

print_info(ten = "Pi", tuoi = "21", thanh_pho= "Ho Chi Minh")
```

```
ten:Pi
tuoi:21
thanh_pho:Ho Chi Minh
```

In [ ]:
```python
# Bài 11: Nêu sự khác biệt giữa *args và **kwargs.
```

Bài 11: Nêu sự khác biệt giữa *args và* *kwargs.

`*args`:

- Sử dụng để cho phép hàm chấp nhận một số lượng **đối số vị trí (positional arguments)** tùy ý.
- Các đối số vị trí được truyền vào sẽ được thu thập lại thành một **tuple**.
- Tên `args` chỉ là quy ước, có thể sử dụng tên khác nhưng phải có dấu `*` đứng trước (ví dụ: `*numbers`).

`kwargs`:**

- Sử dụng để cho phép hàm chấp nhận một số lượng **đối số từ khóa (keyword arguments)** tùy ý.
- Các đối số từ khóa được truyền vào sẽ được thu thập lại thành một **dictionary**.
- Tên `kwargs` chỉ là quy ước, có thể sử dụng tên khác nhưng phải có dấu `**` đứng trước (ví dụ: `**data`).

Tóm lại:

- `*args` dùng cho các đối số theo vị trí không có tên.
- `**kwargs` dùng cho các đối số theo tên (từ khóa) có tên.

In [19]:
```python
# Bài 12: Viết ví dụ sử dụng đồng thời *args và **kwargs.
def example_args_kwargs(arg1, arg2, *args, **kwargs):
  """ Ví dụ hàm sử dụng đồng thời positional, *args và **kwargs """
  print(f"Đối số bắt buộc 1: {arg1}")
  print(f"Đối số bắt buộc 2: {arg2}")
  print(f"*args (đối số vị trí tùy ý): {args}")
  print(f"**kwargs (đối số từ khóa tùy ý): {kwargs}")

# Ví dụ sử dụng
example_args_kwargs(10, 20, 30, 40, 50, name="Alice", age=30, city="New York")

print("\n---")

example_args_kwargs("apple", "banana", color="red", shape="round")
```

```
Đối số bắt buộc 1: 10
Đối số bắt buộc 2: 20
*args (đối số vị trí tùy ý): (30, 40, 50)
**kwargs (đối số từ khóa tùy ý): {'name': 'Alice', 'age': 30, 'city': 'New York'}

---
Đối số bắt buộc 1: apple
Đối số bắt buộc 2: banana
*args (đối số vị trí tùy ý): ()
**kwargs (đối số từ khóa tùy ý): {'color': 'red', 'shape': 'round'}
```

In [20]:
```python
# Bài 13: Định nghĩa biến toàn cục và biến cục bộ, minh họa bằng code.
bien_toan_cuc = "Đây là biến toàn cục"

def in_bien():
  bien_cuc_bo = "Đây là biến cục bộ"
  print("Trong hàm:")
  print(bien_toan_cuc)
  print(bien_cuc_bo)

in_bien()

print("\nNgoài hàm:")
print(bien_toan_cuc)
# print(bien_cuc_bo) # Dòng này sẽ gây lỗi vì bien_cuc_bo chỉ tồn tại trong hàm
```

```
Trong hàm:
Đây là biến toàn cục
Đây là biến cục bộ

Ngoài hàm:
Đây là biến toàn cục
```

Bài 14: Giải thích phạm vi của biến được khai báo trong hàm.

Phạm vi (scope) của biến trong Python xác định nơi mà biến đó có thể được truy cập. Biến được khai báo bên trong một hàm có phạm vi là **cục bộ (local scope)**. Điều này có nghĩa là:

- Biến đó chỉ tồn tại và có thể được truy cập **bên trong hàm** nơi nó được khai báo.
- Khi hàm kết thúc thực thi, biến cục bộ đó sẽ bị hủy.
- Không thể truy cập biến cục bộ từ bên ngoài hàm.

Ví dụ minh họa đã có ở Bài 13, biến `bien_cuc_bo` chỉ có thể truy cập bên trong hàm `in_bien()`.

In [21]:
```
# Bài 15: Viết hàm sử dụng từ khóa global.
bien_toan_cuc_global = 10

def thay_doi_bien_toan_cuc():
  global bien_toan_cuc_global
  bien_toan_cuc_global = 20
  print(f"Trong hàm, biến toàn cục là: {bien_toan_cuc_global}")

print(f"Trước khi gọi hàm, biến toàn cục là: {bien_toan_cuc_global}")
thay_doi_bien_toan_cuc()
print(f"Sau khi gọi hàm, biến toàn cục là: {bien_toan_cuc_global}")
```

```
Trước khi gọi hàm, biến toàn cục là: 10
Trong hàm, biến toàn cục là: 20
Sau khi gọi hàm, biến toàn cục là: 20
```

In [22]:
```
# Bài 16: Viết hàm sử dụng từ khóa nonlocal.
def outer_function():
  bien_bao_ngoai = "Biến của hàm bao ngoài"

  def inner_function():
    nonlocal bien_bao_ngoai
    bien_bao_ngoai = "Biến của hàm bao ngoài đã thay đổi"
    print(f"Trong hàm nội, biến bao ngoài là: {bien_bao_ngoai}")

  print(f"Trước khi gọi hàm nội, biến bao ngoài là: {bien_bao_ngoai}")
  inner_function()
  print(f"Sau khi gọi hàm nội, biến bao ngoài là: {bien_bao_ngoai}")

outer_function()
```

```
Trước khi gọi hàm nội, biến bao ngoài là: Biến của hàm bao ngoài
Trong hàm nội, biến bao ngoài là: Biến của hàm bao ngoài đã thay đổi
Sau khi gọi hàm nội, biến bao ngoài là: Biến của hàm bao ngoài đã thay đổi
```

In [23]:
```
# Bài 17: Cho ví dụ về function annotation để chỉ định kiểu tham số là int.
def cong_hai_so(a: int, b: int) -> int:
  return a + b

print(cong_hai_so(5, 10))
```

```
15
```

In [24]:
```
# Bài 18: Viết một hàm có annotation đầu vào kiểu str và đầu ra kiểu int.
def do_dai_chuoi(chuoi: str) -> int:
```

```
  return len(chuoi)

print(do_dai_chuoi("Hello"))
```

```
5
```

In [25]:
```
# Bài 19: Cho biết print(add.__annotations__) sẽ in ra gì.
# Giả sử hàm add đã được định nghĩa và có annotation
# Nếu chưa có, ta định nghĩa lại hàm add với annotation
def add(a: int, b: int) -> int:
  return a + b

print(add.__annotations__)
```

```
{'a': <class 'int'>, 'b': <class 'int'>, 'return': <class 'int'>}
```

Bài 20: Giải thích ý nghĩa của function annotation.

Function annotation trong Python là các gợi ý về kiểu dữ liệu (type hints) được thêm vào các tham số của hàm và giá trị trả về. Ý nghĩa chính của function annotation là:

- **Tăng khả năng đọc và hiểu code:** Giúp người đọc code dễ dàng biết được kiểu dữ liệu mong đợi của các tham số và kiểu dữ liệu trả về của hàm.
- **Hỗ trợ công cụ phân tích tĩnh (static analysis tools) và trình kiểm tra kiểu (type checkers):** Các công cụ như MyPy có thể sử dụng annotation để kiểm tra lỗi kiểu dữ liệu trước khi chạy code, giúp phát hiện sớm các lỗi tiềm ẩn.
- **Tài liệu hóa code:** Annotation đóng vai trò như một hình thức tài liệu hóa, mô tả rõ ràng hơn về giao diện của hàm.
- **Không ảnh hưởng đến thời gian chạy (runtime):** Annotation không thay đổi cách thức hoạt động của chương trình khi chạy. Chúng chỉ là siêu dữ liệu (metadata) có thể được truy cập thông qua thuộc tính `__annotations__` của hàm.

Tóm lại, function annotation giúp làm cho code Python trở nên rõ ràng, dễ bảo trì và ít lỗi hơn.

In [26]:
```
# Bài 21: Định nghĩa một hàm multiply(x, y=1) có tham số mặc định.
def multiply(x, y=1):
  return x * y

print(multiply(5)) # Sử dụng giá trị mặc định của y
print(multiply(5, 3)) # Ghi đè giá trị mặc định của y
```

```
5
15
```

In [27]:
```
# Bài 22: Viết ví dụ minh họa việc gọi hàm với positional-only và keyword-only a
# cùng lúc.
def combined_args(pos_only, /, keyword_only):
  print(f"Tham số chỉ vị trí: {pos_only}")
  print(f"Tham số chỉ từ khóa: {keyword_only}")

# Gọi hàm
combined_args(10, keyword_only=20)
# combined_args(pos_only=10, keyword_only=20) # Lỗi vì pos_only là positional-on
# combined_args(10, 20) # Lỗi vì keyword_only là keyword-only
```

```
Tham số chỉ vị trí: 10
Tham số chỉ từ khóa: 20
```

In [28]:
```python
# Bài 23: Tạo một hàm nhận danh sách số và trả về số lớn nhất.
def tim_so_lon_nhat(danh_sach_so):
  """ Nhận danh sách số và trả về số lớn nhất """
  if not danh_sach_so:
    return None # Trả về None nếu danh sách rỗng
  so_lon_nhat = danh_sach_so[0]
  for so in danh_sach_so:
    if so > so_lon_nhat:
      so_lon_nhat = so
  return so_lon_nhat

# Ví dụ sử dụng
numbers1 = [1, 5, 2, 8, 3]
print(f"Số lớn nhất trong {numbers1} là: {tim_so_lon_nhat(numbers1)}")

numbers2 = []
print(f"Số lớn nhất trong danh sách rỗng là: {tim_so_lon_nhat(numbers2)}")
```

```
Số lớn nhất trong [1, 5, 2, 8, 3] là: 8
Số lớn nhất trong danh sách rỗng là: None
```

In [31]:
```python
# Bài 24: Viết hàm factorial(n) sử dụng đệ quy.
def factorial(n):
  """ Hàm tính n giai thừa bằng đệ quy!"""
  if n < 0:
    raise ValueError("Giai thừa không định nghĩa cho số âm")
  elif n == 0:
    return 1
  else:
    return n * factorial(n-1)

# Ví dụ sử dụng
print("5 giai thừa là: ", factorial(5))
print("0 giai thừa là: ", factorial(0))
print("-1 giai thừa là: ", factorial(-1))
```

```
5 giai thừa là:  120
0 giai thừa là:  1
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-1926591502.py in <cell line: 0>()
     12 print("5 giai thừa là: ", factorial(5))
     13 print("0 giai thừa là: ", factorial(0))
---> 14 print("-1 giai thừa là: ", factorial(-1))

/tmp/ipython-input-1926591502.py in factorial(n)
      3   """ Hàm tính n giai thừa bằng đệ quy!"""
      4   if n < 0:
----> 5     raise ValueError("Giai thừa không định nghĩa cho số âm")
      6   elif n == 0:
      7     return 1

ValueError: Giai thừa không định nghĩa cho số âm
```

In [30]:
```python
# Bài 25: Cho biết kết quả của đoạn code:
# def f(x, y=2): return x*y
# print(f(3))
```

```
# Kết quả của đoạn code trên là 6.
# Giải thích:
# Hàm f nhận hai đối số x và y, với y có giá trị mặc định là 2.
# Khi gọi f(3), giá trị 3 được gán cho x, và y sử dụng giá trị mặc định là 2.
# Hàm trả về kết quả của x * y, tức là 3 * 2 = 6.
# Do đó, print(f(3)) sẽ in ra 6.
```

In [32]:
```
# Bài 26: Giải thích tại sao dùng None làm giá trị mặc định đôi khi tốt hơn danh

# Sử dụng None làm giá trị mặc định cho các tham số kiểu danh sách (list) hoặc d
# đôi khi tốt hơn việc sử dụng trực tiếp một danh sách rỗng [] hoặc dictionary r
# Lý do là bởi vì các giá trị mặc định trong Python chỉ được tạo ra một lần khi
# không phải mỗi lần hàm được gọi.

# Nếu bạn sử dụng một danh sách rỗng làm giá trị mặc định và gọi hàm nhiều lần,
# tất cả các lần gọi hàm sẽ sử dụng cùng một danh sách rỗng đó.
# Điều này có thể dẫn đến hành vi không mong muốn khi bạn thêm hoặc sửa đổi danh
# vì những thay đổi này sẽ ảnh hưởng đến tất cả các lần gọi hàm sau này.

# Ví dụ minh họa vấn đề khi dùng list rỗng làm default:
def append_to_list_bad(value, my_list=[]):
  my_list.append(value)
  return my_list

print("Ví dụ dùng list rỗng làm default:")
print(append_to_list_bad(1)) # Output: [1]
print(append_to_list_bad(2)) # Output: [1, 2] - Kết quả không mong muốn!

# Ví dụ minh họa cách dùng None làm default (cách đúng):
def append_to_list_good(value, my_list=None):
  if my_list is None:
    my_list = []
  my_list.append(value)
  return my_list

print("\nVí dụ dùng None làm default:")
print(append_to_list_good(1)) # Output: [1]
print(append_to_list_good(2)) # Output: [2] - Mỗi lần gọi là một danh sách mới
```

```
Ví dụ dùng list rỗng làm default:
[1]
[1, 2]

Ví dụ dùng None làm default:
[1]
[2]
```

In [33]:
```
# Bài 27: Viết hàm kiểm tra một số có phải số nguyên tố hay không.

def is_prime(n):
  """
  Kiểm tra xem một số có phải là số nguyên tố hay không.
  Số nguyên tố là số tự nhiên lớn hơn 1 và chỉ có hai ước dương phân biệt là 1 v
  """
  if n <= 1:
    return False
  if n <= 3:
    return True
  if n % 2 == 0 or n % 3 == 0:
    return False
```

```
  i = 5
  while i * i <= n:
    if n % i == 0 or n % (i + 2) == 0:
      return False
    i += 6
  return True

# Ví dụ sử dụng
print(f"11 là số nguyên tố: {is_prime(11)}")
print(f"15 là số nguyên tố: {is_prime(15)}")
```

```
11 là số nguyên tố: True
15 là số nguyên tố: False
```

In [34]:
```
# Bài 28: Cho ví dụ về việc sử dụng hàm như một đối số truyền vào hàm khác.

def ap_dung_ham(func, danh_sach):
  """
  Hàm nhận một hàm khác (func) và một danh sách (danh_sach)
  và áp dụng hàm đó cho từng phần tử của danh sách.
  """
  ket_qua = []
  for phan_tu in danh_sach:
    ket_qua.append(func(phan_tu))
  return ket_qua

# Hàm ví dụ để truyền vào
def binh_phuong(x):
  return x * x

def nhan_doi(x):
  return x * 2

# Sử dụng hàm ap_dung_ham với hàm binh_phuong
numbers = [1, 2, 3, 4, 5]
binh_phuong_cac_so = ap_dung_ham(binh_phuong, numbers)
print(f"Bình phương của {numbers} là: {binh_phuong_cac_so}")

# Sử dụng hàm ap_dung_ham với hàm nhan_doi
nhan_doi_cac_so = ap_dung_ham(nhan_doi, numbers)
print(f"Nhân đôi của {numbers} là: {nhan_doi_cac_so}")
```

```
Bình phương của [1, 2, 3, 4, 5] là: [1, 4, 9, 16, 25]
Nhân đôi của [1, 2, 3, 4, 5] là: [2, 4, 6, 8, 10]
```

In [35]:
```
# Bài 29: Viết hàm ẩn danh (lambda) để tính bình phương một số.

binh_phuong_lambda = lambda x: x ** 2

print(f"Bình phương của 7 là: {binh_phuong_lambda(7)}")
print(f"Bình phương của 10 là: {binh_phuong_lambda(10)}")
```

```
Bình phương của 7 là: 49
Bình phương của 10 là: 100
```

In [36]:
```
# Bài 30: Phân biệt return và print trong hàm Python.

# return:
# - Kết thúc việc thực thi của hàm và trả về một giá trị từ hàm đó.
# - Khi return được thực thi, hàm dừng lại ngay lập tức.
```

```
# - Giá trị trả về có thể được sử dụng (gán cho biến, dùng trong biểu thức...).
# - Nếu không có return, hàm mặc định trả về None.

# print:
# - Chỉ đơn thuần là in (hiển thị) một giá trị ra console hoặc output.
# - Không kết thúc việc thực thi của hàm.
# - Giá trị được in ra không được trả về từ hàm để sử dụng tiếp.

# Ví dụ minh họa:
def ham_return(x):
  return x * 2 # Trả về giá trị

def ham_print(x):
  print(x * 2) # Chỉ in giá trị

print("Sử dụng hàm return:")
ket_qua_return = ham_return(5)
print(f"Giá trị nhận được từ hàm return: {ket_qua_return}") # Có thể sử dụng giá

print("\nSử dụng hàm print:")
ket_qua_print = ham_print(5) # Chỉ in ra 10
print(f"Giá trị nhận được từ hàm print: {ket_qua_print}") # Giá trị nhận được là
```

```
Sử dụng hàm return:
Giá trị nhận được từ hàm return: 10

Sử dụng hàm print:
10
Giá trị nhận được từ hàm print: None
```

Phần B

In [39]:

```
# Bài 1: Viết một hàm calculate_grade(score) trả về hạng (A, B, C, D, F) dựa trê
def calculate_grade(score):
  """Trả về hạng dựa trên điểm số."""
  if 9 <= score <= 10:
    return 'A'
  elif 8 <= score < 9:
    return 'B'
  elif 7 <= score < 8:
    return 'C'
  elif 6 <= score < 7:
    return 'D'
  elif 0 <= score < 6:
    return 'F'
  else:
    return "Điểm không hợp lệ"

# Ví dụ sử dụng
print(f"Điểm 9.5: {calculate_grade(9.5)}")
print(f"Điểm 8.2: {calculate_grade(8.2)}")
print(f"Điểm 6.8: {calculate_grade(6.8)}")
print(f"Điểm 10.5: {calculate_grade(10.5)}")
```

```
Điểm 9.5: A
Điểm 8.2: B
Điểm 6.8: D
Điểm 10.5: Điểm không hợp lệ
```

In [40]:
```python
# Bài 2: Tạo hàm area(shape, **kwargs) tính diện tích hình chữ nhật, tam giác ho
import math

def area(shape, **kwargs):
  """Tính diện tích các hình dựa vào tên hình và tham số."""
  if shape.lower() == 'rectangle':
    # Diện tích hình chữ nhật: width * height
    if 'width' in kwargs and 'height' in kwargs:
      return kwargs['width'] * kwargs['height']
    else:
      return "Thiếu tham số 'width' hoặc 'height' cho hình chữ nhật."
  elif shape.lower() == 'triangle':
    # Diện tích tam giác: 0.5 * base * height
    if 'base' in kwargs and 'height' in kwargs:
      return 0.5 * kwargs['base'] * kwargs['height']
    else:
      return "Thiếu tham số 'base' hoặc 'height' cho tam giác."
  elif shape.lower() == 'circle':
    # Diện tích hình tròn: pi * radius^2
    if 'radius' in kwargs:
      return math.pi * (kwargs['radius'] ** 2)
    else:
      return "Thiếu tham số 'radius' cho hình tròn."
  else:
    return "Hình không hỗ trợ."

# Ví dụ sử dụng
print(f"Diện tích hình chữ nhật (4x5): {area('rectangle', width=4, height=5)}")
print(f"Diện tích tam giác (đáy 6, cao 3): {area('triangle', base=6, height=3)}"
print(f"Diện tích hình tròn (bán kính 7): {area('circle', radius=7)}")
print(f"Diện tích hình vuông (cạnh 5): {area('square', side=5)}") # Hình không h
print(f"Diện tích hình chữ nhật thiếu tham số: {area('rectangle', width=4)}")
```

```
Diện tích hình chữ nhật (4x5): 20
Diện tích tam giác (đáy 6, cao 3): 9.0
Diện tích hình tròn (bán kính 7): 153.93804002589985
Diện tích hình vuông (cạnh 5): Hình không hỗ trợ.
Diện tích hình chữ nhật thiếu tham số: Thiếu tham số 'width' hoặc 'height' cho hì
nh chữ nhật.
```

In [41]:
```python
# Bài 3: Viết hàm count_words(text) trả về số từ trong một chuỗi.
import re

def count_words(text):
  """Đếm số từ trong một chuỗi."""
  # Sử dụng biểu thức chính quy để tìm các từ (chuỗi ký tự chữ cái)
  words = re.findall(r'\b\w+\b', text.lower())
  return len(words)

# Ví dụ sử dụng
text1 = "Đây là một câu văn đơn giản."
print(f"Số từ trong '{text1}': {count_words(text1)}")

text2 = "Hello world! This is a test."
print(f"Số từ trong '{text2}': {count_words(text2)}")

text3 = ""
print(f"Số từ trong '{text3}': {count_words(text3)}")
```

```
Số từ trong 'Đây là một câu văn đơn giản.': 7
Số từ trong 'Hello world! This is a test.': 6
Số từ trong '': 0
```

In [42]:
```python
# Bài 4: Định nghĩa hàm fibonacci(n) trả về dãy Fibonacci có n phần tử.
def fibonacci(n):
  """Trả về danh sách chứa n phần tử đầu tiên của dãy Fibonacci."""
  if n <= 0:
    return []
  elif n == 1:
    return [0]
  else:
    dãy = [0, 1]
    while len(dãy) < n:
      next_fib = dãy[-1] + dãy[-2]
      dãy.append(next_fib)
    return dãy

# Ví dụ sử dụng
print(f"Dãy Fibonacci 0 phần tử: {fibonacci(0)}")
print(f"Dãy Fibonacci 1 phần tử: {fibonacci(1)}")
print(f"Dãy Fibonacci 5 phần tử: {fibonacci(5)}")
print(f"Dãy Fibonacci 10 phần tử: {fibonacci(10)}")
```

```
Dãy Fibonacci 0 phần tử: []
Dãy Fibonacci 1 phần tử: [0]
Dãy Fibonacci 5 phần tử: [0, 1, 1, 2, 3]
Dãy Fibonacci 10 phần tử: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

In [43]:
```python
# Bài 5: Viết hàm is_palindrome(s) kiểm tra chuỗi có đối xứng không.
import re

def is_palindrome(s):
  """Kiểm tra xem chuỗi có đối xứng không (không phân biệt hoa thường và dấu các
  # Chuyển chuỗi về chữ thường và loại bỏ các ký tự không phải chữ cái hoặc số
  chuoi_sach = re.sub(r'[^a-z0-9]', '', s.lower())
  # So sánh chuỗi đã làm sạch với chuỗi đảo ngược của nó
  return chuoi_sach == chuoi_sach[::-1]

# Ví dụ sử dụng
print(f"'madam' là chuỗi đối xứng: {is_palindrome('madam')}")
print(f"'racecar' là chuỗi đối xứng: {is_palindrome('racecar')}")
print(f"'hello' là chuỗi đối xứng: {is_palindrome('hello')}")
print(f"'A man, a plan, a canal: Panama.' là chuỗi đối xứng: {is_palindrome('A m
```

```
'madam' là chuỗi đối xứng: True
'racecar' là chuỗi đối xứng: True
'hello' là chuỗi đối xứng: False
'A man, a plan, a canal: Panama.' là chuỗi đối xứng: True
```

In [45]:
```python
# Bài 6: Viết hàm nhận vào danh sách số, trả về danh sách mới chứa bình phương c
def binh_phuong_so_le(danh_sach_so):
  """Nhận danh sách số, trả về danh sách bình phương các số lẻ."""
  ket_qua = []
  for so in danh_sach_so:
    if so % 2 != 0: # Kiểm tra nếu là số lẻ
      ket_qua.append(so ** 2)
  return ket_qua

# Ví dụ sử dụng
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
binh_phuong_le = binh_phuong_so_le(numbers)
print(f"Bình phương các số lẻ trong {numbers} là: {binh_phuong_le}")

empty_list = []
print(f"Bình phương các số lẻ trong {empty_list} là: {binh_phuong_so_le(empty_li
```

```
Bình phương các số lẻ trong [1, 2, 3, 4, 5, 6, 7, 8, 9] là: [1, 9, 25, 49, 81]
Bình phương các số lẻ trong [] là: []
```

In [46]:
```
# Bài 7: Viết hàm sort_numbers(*args, reverse=False) sắp xếp các số theo thứ tự
def sort_numbers(*args, reverse=False):
  """Sắp xếp các số truyền vào theo thứ tự tăng dần hoặc giảm dần."""
  # Chuyển args thành danh sách để có thể sắp xếp
  numbers = list(args)
  numbers.sort(reverse=reverse)
  return numbers

# Ví dụ sử dụng
sorted_asc = sort_numbers(5, 2, 8, 1, 9)
print(f"Sắp xếp tăng dần: {sorted_asc}")

sorted_desc = sort_numbers(5, 2, 8, 1, 9, reverse=True)
print(f"Sắp xếp giảm dần: {sorted_desc}")

single_number = sort_numbers(10)
print(f"Sắp xếp một số: {single_number}")

no_number = sort_numbers()
print(f"Sắp xếp không số: {no_number}")
```

```
Sắp xếp tăng dần: [1, 2, 5, 8, 9]
Sắp xếp giảm dần: [9, 8, 5, 2, 1]
Sắp xếp một số: [10]
Sắp xếp không số: []
```

In [48]:
```
# Bài 8: Viết hàm student_info(name, age, **kwargs) để quản lý thông tin sinh vi
def student_info(name, age, **kwargs):
  """Quản lý thông tin sinh viên."""
  print(f"Tên: {name}")
  print(f"Tuổi: {age}")
  if kwargs:
    print("Thông tin bổ sung:")
    for key, value in kwargs.items():
      print(f"- {key.replace('_', ' ').title()}: {value}") # Format key cho dễ đ

# Ví dụ sử dụng
student_info("Nguyen Van A", 20, major="DS", university="HCMUS")
print("-" * 20)
student_info("Tran Thi B", 21, hometown="Hanoi")
```

```
Tên: Nguyen Van A
Tuổi: 20
Thông tin bổ sung:
- Major: DS
- University: HCMUS
--------------------
Tên: Tran Thi B
Tuổi: 21
Thông tin bổ sung:
- Hometown: Hanoi
```

In [49]:
```python
# Bài 9: Viết hàm apply_operation(x, y, func) nhận một hàm làm đối số để thực hi
def apply_operation(x, y, func):
  """Áp dụng một hàm (phép toán) lên hai số x và y."""
  return func(x, y)

# Các hàm phép toán
def add(a, b):
  return a + b

def subtract(a, b):
  return a - b

def multiply(a, b):
  return a * b

# Ví dụ sử dụng
result_add = apply_operation(10, 5, add)
print(f"Áp dụng phép cộng cho 10 và 5: {result_add}")

result_subtract = apply_operation(10, 5, subtract)
print(f"Áp dụng phép trừ cho 10 và 5: {result_subtract}")

result_multiply = apply_operation(10, 5, multiply)
print(f"Áp dụng phép nhân cho 10 và 5: {result_multiply}")
```

```
Áp dụng phép cộng cho 10 và 5: 15
Áp dụng phép trừ cho 10 và 5: 5
Áp dụng phép nhân cho 10 và 5: 50
```

In [50]:
```python
# Bài 10: Viết hàm safe_divide(a, b) xử lý ngoại lệ khi chia cho 0.
def safe_divide(a, b):
  """Chia hai số a cho b, xử lý trường hợp chia cho 0."""
  try:
    result = a / b
    return result
  except ZeroDivisionError:
    return "Lỗi: Không thể chia cho 0."

# Ví dụ sử dụng
print(f"10 / 2 = {safe_divide(10, 2)}")
print(f"10 / 0 = {safe_divide(10, 0)}")
print(f"5 / 3 = {safe_divide(5, 3)}")
```

```
10 / 2 = 5.0
10 / 0 = Lỗi: Không thể chia cho 0.
5 / 3 = 1.6666666666666667
```

In [51]:
```python
# Bài 11: Viết hàm word_frequency(text) trả về từ điển đếm số lần xuất hiện của
import re
from collections import Counter

def word_frequency(text):
  """Trả về từ điển đếm số lần xuất hiện của mỗi từ trong chuỗi."""
  # Chuyển chuỗi về chữ thường và tìm tất cả các từ
  words = re.findall(r'\b\w+\b', text.lower())
  # Sử dụng Counter để đếm tần suất xuất hiện của các từ
  return Counter(words)

# Ví dụ sử dụng
text = "Python programming is fun. Python is versatile."
```

```
word_counts = word_frequency(text)
print(f"Tần suất từ trong '{text}': {word_counts}")

text2 = "Hello world hello python"
print(f"Tần suất từ trong '{text2}': {word_frequency(text2)}")
```

```
Tần suất từ trong 'Python programming is fun. Python is versatile.': Counter({'py
thon': 2, 'is': 2, 'programming': 1, 'fun': 1, 'versatile': 1})
Tần suất từ trong 'Hello world hello python': Counter({'hello': 2, 'world': 1, 'p
ython': 1})
```

In [52]:
```
# Bài 12: Viết hàm nhận vào danh sách số, trả về giá trị trung bình, lớn nhất, n
def analyze_numbers(danh_sach_so):
  """Nhận danh sách số và trả về giá trị trung bình, lớn nhất, nhỏ nhất."""
  if not danh_sach_so:
    return None, None, None # Trả về None nếu danh sách rỗng

  trung_binh = sum(danh_sach_so) / len(danh_sach_so)
  lon_nhat = max(danh_sach_so)
  nho_nhat = min(danh_sach_so)

  return trung_binh, lon_nhat, nho_nhat

# Ví dụ sử dụng
numbers = [10, 20, 30, 40, 50]
trung_binh, lon_nhat, nho_nhat = analyze_numbers(numbers)
print(f"Danh sách: {numbers}")
print(f"Trung bình: {trung_binh}")
print(f"Lớn nhất: {lon_nhat}")
print(f"Nhỏ nhất: {nho_nhat}")

empty_list = []
trung_binh_empty, lon_nhat_empty, nho_nhat_empty = analyze_numbers(empty_list)
print(f"\nDanh sách rỗng: {empty_list}")
print(f"Trung bình: {trung_binh_empty}")
print(f"Lớn nhất: {lon_nhat_empty}")
print(f"Nhỏ nhất: {nho_nhat_empty}")
```

```
Danh sách: [10, 20, 30, 40, 50]
Trung bình: 30.0
Lớn nhất: 50
Nhỏ nhất: 10

Danh sách rỗng: []
Trung bình: None
Lớn nhất: None
Nhỏ nhất: None
```

In [53]:
```
# Bài 13: Viết hàm to_upper(strings) nhận danh sách chuỗi và trả về danh sách ch
def to_upper(strings):
  """Nhận danh sách chuỗi và trả về danh sách chuỗi viết hoa."""
  return [s.upper() for s in strings]

# Ví dụ sử dụng
string_list = ["hello", "world", "python"]
upper_list = to_upper(string_list)
print(f"Danh sách gốc: {string_list}")
print(f"Danh sách viết hoa: {upper_list}")

empty_list = []
upper_empty = to_upper(empty_list)
```

```
print(f"Danh sách rỗng: {empty_list}")
print(f"Danh sách viết hoa (rỗng): {upper_empty}")
```

```
Danh sách gốc: ['hello', 'world', 'python']
Danh sách viết hoa: ['HELLO', 'WORLD', 'PYTHON']
Danh sách rỗng: []
Danh sách viết hoa (rỗng): []
```

In [54]:
```
# Bài 14: Viết hàm đệ quy tính tổng các số từ 1 đến n.
def sum_recursive(n):
  """Tính tổng các số từ 1 đến n bằng đệ quy."""
  if n <= 0:
    return 0
  else:
    return n + sum_recursive(n - 1)

# Ví dụ sử dụng
print(f"Tổng từ 1 đến 5: {sum_recursive(5)}")
print(f"Tổng từ 1 đến 0: {sum_recursive(0)}")
print(f"Tổng từ 1 đến 10: {sum_recursive(10)}")
```

```
Tổng từ 1 đến 5: 15
Tổng từ 1 đến 0: 0
Tổng từ 1 đến 10: 55
```

In [55]:
```
# Bài 15: Viết hàm compose(f, g) trả về hàm hợp của f và g.
def compose(f, g):
  """Trả về hàm hợp của f và g, tức là h(x) = f(g(x))."""
  def h(x):
    return f(g(x))
  return h

# Các hàm ví dụ
def cong_mot(x):
  return x + 1

def binh_phuong(x):
  return x ** 2

# Tạo hàm hợp: binh_phuong(cong_mot(x))
ham_hop = compose(binh_phuong, cong_mot)

# Ví dụ sử dụng
# ham_hop(2) sẽ thực hiện binh_phuong(cong_mot(2)) = binh_phuong(3) = 9
print(f"Kết quả hàm hợp của bình phương và cộng 1 cho số 2: {ham_hop(2)}")

# Tạo hàm hợp: cong_mot(binh_phuong(x))
ham_hop2 = compose(cong_mot, binh_phuong)
# ham_hop2(2) sẽ thực hiện cong_mot(binh_phuong(2)) = cong_mot(4) = 5
print(f"Kết quả hàm hợp của cộng 1 và bình phương cho số 2: {ham_hop2(2)}")
```

```
Kết quả hàm hợp của bình phương và cộng 1 cho số 2: 9
Kết quả hàm hợp của cộng 1 và bình phương cho số 2: 5
```

In [56]:
```
# Bài 16: Viết hàm timer(func) đo thời gian chạy của một hàm.
import time

def timer(func):
  """Đo thời gian chạy của một hàm và trả về kết quả của hàm đó."""
  def wrapper(*args, **kwargs):
    start_time = time.time()
```

```
    result = func(*args, **kwargs)
    end_time = time.time()
    duration = end_time - start_time
    print(f"Hàm '{func.__name__}' chạy mất {duration:.6f} giây.")
    return result
  return wrapper

# Ví dụ sử dụng decorator @timer
@timer
def slow_function(delay):
  """Một hàm mô phỏng công việc tốn thời gian."""
  time.sleep(delay)
  return "Done sleeping"

@timer
def fast_function(a, b):
  """Một hàm chạy nhanh."""
  return a + b

# Chạy các hàm đã được "timer"
slow_function(1)
print(fast_function(3, 4))
```

```
Hàm 'slow_function' chạy mất 1.000139 giây.
Hàm 'fast_function' chạy mất 0.000002 giây.
7
```

In [57]:
```
# Bài 17: Viết hàm greet(name: str, times: int) -> str có annotation mô tả kiểu.
def greet(name: str, times: int) -> str:
  """Hàm chào một người nhiều lần với annotation kiểu."""
  return (f"Xin chào, {name}!" + " ") * times

print(greet("Lan", 3))
print(greet("Python", 1))

print(greet(123, 2))
```

```
Xin chào, Lan! Xin chào, Lan! Xin chào, Lan!
Xin chào, Python!
Xin chào, 123! Xin chào, 123!
```

In [58]:
```
# Bài 18: Viết hàm make_counter() trả về một hàm dùng nonlocal để đếm số lần đượ
def make_counter():
  """Trả về một hàm closure dùng nonlocal để đếm số lần được gọi."""
  count = 0

  def counter():
    nonlocal count
    count += 1
    return count
  return counter

my_counter = make_counter()

print(f"Gọi lần 1: {my_counter()}") # Output: 1
print(f"Gọi lần 2: {my_counter()}") # Output: 2
print(f"Gọi lần 3: {my_counter()}") # Output: 3

another_counter = make_counter() # Tạo một counter mới
print(f"\nGọi counter mới lần 1: {another_counter()}") # Output: 1
```

```
Gọi lần 1: 1
Gọi lần 2: 2
Gọi lần 3: 3

Gọi counter mới lần 1: 1
```

In [59]:

```python
# Bài 19: Viết chương trình định nghĩa nhiều hàm để quản lý danh bạ (thêm, xóa,

# Sử dụng một từ điển để lưu danh bạ: {tên: số_điện_thoại}
danh_ba = {}

def them_lien_lac(ten, so_dien_thoai):
  """Thêm một liên lạc mới vào danh bạ."""
  if ten in danh_ba:
    print(f"Liên lạc '{ten}' đã tồn tại trong danh bạ.")
  else:
    danh_ba[ten] = so_dien_thoai
    print(f"Đã thêm liên lạc '{ten}' với số điện thoại {so_dien_thoai}.")

def xoa_lien_lac(ten):
  """Xóa một liên lạc khỏi danh bạ."""
  if ten in danh_ba:
    del danh_ba[ten]
    print(f"Đã xóa liên lạc '{ten}' khỏi danh bạ.")
  else:
    print(f"Không tìm thấy liên lạc '{ten}' trong danh bạ.")

def tim_kiem_lien_lac(ten):
  """Tìm kiếm số điện thoại của một liên lạc."""
  if ten in danh_ba:
    print(f"Số điện thoại của '{ten}': {danh_ba[ten]}.")
    return danh_ba[ten]
  else:
    print(f"Không tìm thấy liên lạc '{ten}' trong danh bạ.")
    return None

def hien_thi_danh_ba():
    """Hiển thị toàn bộ danh bạ."""
    if danh_ba:
        print("\n--- Danh bạ ---")
        for ten, so in danh_ba.items():
            print(f"{ten}: {so}")
        print("---------------")
    else:
        print("\nDanh bạ trống.")

hien_thi_danh_ba()

them_lien_lac("Alice", "0123456789")
them_lien_lac("Bob", "0987654321")
them_lien_lac("Alice", "0112233445")

hien_thi_danh_ba()

tim_kiem_lien_lac("Alice")
tim_kiem_lien_lac("Charlie")

xoa_lien_lac("Bob")
xoa_lien_lac("David")
```

```
hien_thi_danh_ba()
```

```
Danh bạ trống.
Đã thêm liên lạc 'Alice' với số điện thoại 0123456789.
Đã thêm liên lạc 'Bob' với số điện thoại 0987654321.
Liên lạc 'Alice' đã tồn tại trong danh bạ.

--- Danh bạ ---
Alice: 0123456789
Bob: 0987654321
---------------
Số điện thoại của 'Alice': 0123456789.
Không tìm thấy liên lạc 'Charlie' trong danh bạ.
Đã xóa liên lạc 'Bob' khỏi danh bạ.
Không tìm thấy liên lạc 'David' trong danh bạ.

--- Danh bạ ---
Alice: 0123456789
---------------
```

In [60]:
```python
# Bài 20: Viết hàm validate_password(password) kiểm tra tính hợp lệ của mật khẩu
# các tiêu chí (độ dài, ký tự đặc biệt, chữ số, chữ cái).
import re

def validate_password(password):
  """
  Kiểm tra tính hợp lệ của mật khẩu theo các tiêu chí:
  - Độ dài tối thiểu 8 ký tự.
  - Chứa ít nhất một chữ cái viết hoa.
  - Chứa ít nhất một chữ cái viết thường.
  - Chứa ít nhất một chữ số.
  - Chứa ít nhất một ký tự đặc biệt (@$!%*?&).
  """
  min_length = 8
  has_upper = re.search(r'[A-Z]', password)
  has_lower = re.search(r'[a-z]', password)
  has_digit = re.search(r'[0-9]', password)
  has_special = re.search(r'[@$!%*?&]', password)

  is_valid = True
  messages = []

  if len(password) < min_length:
    is_valid = False
    messages.append(f"- Mật khẩu phải có ít nhất {min_length} ký tự.")
  if not has_upper:
    is_valid = False
    messages.append("- Mật khẩu phải chứa ít nhất một chữ cái viết hoa.")
  if not has_lower:
    is_valid = False
    messages.append("- Mật khẩu phải chứa ít nhất một chữ cái viết thường.")
  if not has_digit:
    is_valid = False
    messages.append("- Mật khẩu phải chứa ít nhất một chữ số.")
  if not has_special:
    is_valid = False
    messages.append("- Mật khẩu phải chứa ít nhất một ký tự đặc biệt (@$!%*?&)."

  if is_valid:
```

```
    print("Mật khẩu hợp lệ.")
    return True
  else:
    print("Mật khẩu không hợp lệ:")
    for msg in messages:
      print(msg)
    return False

# Ví dụ sử dụng
print("Kiểm tra mật khẩu 'Abc@1234':")
validate_password("Abc@1234") # Hợp lệ

print("\nKiểm tra mật khẩu 'password123':")
validate_password("password123") # Thiếu chữ hoa, ký tự đặc biệt

print("\nKiểm tra mật khẩu 'SHORT':")
validate_password("SHORT") # Thiếu độ dài, chữ thường, số, ký tự đặc biệt

print("\nKiểm tra mật khẩu 'Testing!':")
validate_password("Testing!") # Thiếu số

print("\nKiểm tra mật khẩu 'Password123':")
validate_password("Password123") # Thiếu ký tự đặc biệt
```

```
Kiểm tra mật khẩu 'Abc@1234':
Mật khẩu hợp lệ.

Kiểm tra mật khẩu 'password123':
Mật khẩu không hợp lệ:
- Mật khẩu phải chứa ít nhất một chữ cái viết hoa.
- Mật khẩu phải chứa ít nhất một ký tự đặc biệt (@$!%*?&).

Kiểm tra mật khẩu 'SHORT':
Mật khẩu không hợp lệ:
- Mật khẩu phải có ít nhất 8 ký tự.
- Mật khẩu phải chứa ít nhất một chữ cái viết thường.
- Mật khẩu phải chứa ít nhất một chữ số.
- Mật khẩu phải chứa ít nhất một ký tự đặc biệt (@$!%*?&).

Kiểm tra mật khẩu 'Testing!':
Mật khẩu không hợp lệ:
- Mật khẩu phải chứa ít nhất một chữ số.

Kiểm tra mật khẩu 'Password123':
Mật khẩu không hợp lệ:
- Mật khẩu phải chứa ít nhất một ký tự đặc biệt (@$!%*?&).
```

Out[60]: False

Bài tập Thực hành Python - Bài số 5

Phần A

In [61]:
```
# Bài 1: Viết chương trình import module math và in ra giá trị của π.
import math
print(math.pi)
```

```
3.141592653589793
```

In [62]:
```
# Bài 2: Sử dụng hàm sqrt() trong module math để tính căn bậc hai của 49.
import math
print(math.sqrt(49))
```

```
7.0
```

In [63]:
```
# Bài 3: Viết chương trình import module random và in ra một số ngẫu nhiên từ 1
import random
print(random.randint(1, 10))
```

```
3
```

In [64]:
```
# Bài 4: Import module datetime và in ra ngày giờ hiện tại.
import datetime
print(datetime.datetime.now())
```

```
2025-10-18 14:37:11.922784
```

In [65]:
```
# Bài 5: Viết chương trình sử dụng built-in function len() để in độ dài của một
chuoi = "Hello Python"
print(len(chuoi))
```

```
12
```

In [66]:
```
# Bài 6: Sử dụng hàm sum() để tính tổng của list [1,2,3,4,5].
my_list = [1, 2, 3, 4, 5]
print(sum(my_list))
```

```
15
```

In [67]:
```
# Bài 7: Sử dụng hàm min() để tìm số nhỏ nhất trong [3,8,2,5].
my_list = [3, 8, 2, 5]
print(min(my_list))
```

```
2
```

In [68]:
```
# Bài 8: Sử dụng hàm max() để tìm số lớn nhất trong [10,20,5,7].
my_list = [10, 20, 5, 7]
print(max(my_list))
```

```
20
```

In [69]:
```
# Bài 9: Viết chương trình nhập một số từ bàn phím và dùng hàm abs() để in giá t
num = float(input("Nhập một số: "))
print(abs(num))
```

```
Nhập một số: 5
5.0
```

In [70]:
```
# Bài 10: Sử dụng hàm round() để làm tròn số 3.14159 với 2 chữ số thập phân.
print(round(3.14159, 2))
```

```
3.14
```

In [71]:
```
# Bài 11: Viết chương trình sử dụng built-in function type() để in kiểu dữ liệu
my_variable = "Hello"
print(type(my_variable))

my_number = 10
print(type(my_number))
```

```
<class 'str'>
<class 'int'>
```

In [72]:
```
# Bài 12: Sử dụng hàm dir() để in tất cả hàm trong module math.
import math
print(dir(math))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb', 'copysign', 'c
os', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fab
s', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'in
f', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma',
'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow',
'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'sumprod', 'tan', 'tanh',
'tau', 'trunc', 'ulp']
```

In [73]:
```
# Bài 13: Viết chương trình sử dụng built-in function id() để in địa chỉ của một
a = 10
print(id(a))

b = a
print(id(b))
```

```
11654664
11654664
```

In [74]:
```
# Bài 14: Sử dụng built-in function sorted() để sắp xếp list [9,1,5,3].
my_list = [9, 1, 5, 3]
sorted_list = sorted(my_list)
print(sorted_list)
```

```
[1, 3, 5, 9]
```

In [75]:
```
# Bài 15: Viết chương trình dùng hàm help() để hiển thị tài liệu của hàm len.

help(len)
```

```
Help on built-in function len in module builtins:

len(obj, /)
    Return the number of items in a container.
```

In [76]:
```
# Bài 16: Import module os và in ra thư mục hiện tại.
import os
print(os.getcwd())
```

```
/content
```

In [77]:
```
# Bài 17: Viết chương trình sử dụng built-in function pow() để tính 2^5.
print(pow(2, 5))
```

```
32
```

In [78]:
```
# Bài 18: Sử dụng hàm chr() để in ký tự của mã ASCII 65.
print(chr(65))
```

```
A
```

In [79]:
```
# Bài 19: Sử dụng hàm ord() để in mã ASCII của ký tự 'A'.
print(ord('A'))
```

```
65
```

In [80]:
```
# Bài 20: Viết chương trình sử dụng built-in function bin() để đổi số 10 sang nh
print(bin(10))
```

```
0b1010
```

In [81]:
```
# Bài 21: Sử dụng hàm oct() để đổi số 64 sang hệ bát phân.
print(oct(64))
```

```
0o100
```

In [82]:
```
# Bài 22: Sử dụng hàm hex() để đổi số 255 sang hệ thập lục phân.
print(hex(255))
```

```
0xff
```

In [83]:
```
# Bài 23: Viết chương trình sử dụng built-in function any() với list [False,True
print(any([False, True, False]))
```

```
True
```

In [84]:
```
# Bài 24: Sử dụng hàm all() với list [True,True,True].
print(all([True, True, True]))
```

```
True
```

In [85]:
```
# Bài 25: Viết chương trình sử dụng built-in function range() để in các số từ 0
for i in range(10):
  print(i, end=' ')
print()
```

```
0 1 2 3 4 5 6 7 8 9
```

In [86]:
```
# Bài 26: Sử dụng hàm map() để tính bình phương của list [1,2,3,4].
my_list = [1, 2, 3, 4]
squared_list = list(map(lambda x: x**2, my_list))
print(squared_list)
```

```
[1, 4, 9, 16]
```

In [87]:
```
# Bài 27: Sử dụng hàm filter() để lọc ra các số chẵn trong list [1,2,3,4,5,6].
my_list = [1, 2, 3, 4, 5, 6]
even_numbers = list(filter(lambda x: x % 2 == 0, my_list))
print(even_numbers)
```

```
[2, 4, 6]
```

In [88]:
```
# Bài 28: Viết chương trình dùng zip() để gộp hai list thành list các tuple.
list1 = [1, 2, 3]
list2 = ['a', 'b', 'c']
zipped_list = list(zip(list1, list2))
print(zipped_list)
```

```
[(1, 'a'), (2, 'b'), (3, 'c')]
```

In [89]:
```
# Bài 29: Sử dụng hàm enumerate() để duyệt qua list [‘a’,‘b’,‘c’].
my_list = ['a', 'b', 'c']
for index, value in enumerate(my_list):
  print(f"Index: {index}, Value: {value}")
```

```
Index: 0, Value: a
Index: 1, Value: b
Index: 2, Value: c
```

In [90]:
```
# Bài 30: Viết chương trình sử dụng built-in function eval() để tính biểu thức n


expression = input("Nhập một biểu thức toán học: ")

result = eval(expression)
print(f"Kết quả của biểu thức '{expression}' là: {result}")
```

```
Nhập một biểu thức toán học: 7 + 5
Kết quả của biểu thức '7 + 5' là: 12
```

Phần B

In [91]:
```
# Bài 1: Viết một module tên là mycalc.py có các hàm cộng, trừ, nhân, chia.

%%writefile mycalc.py
def add(a, b):
  return a + b

def subtract(a, b):
  return a - b

def multiply(a, b):
  return a * b

def divide(a, b):
  if b == 0:
    return "Lỗi: Không thể chia cho 0"
  return a / b
```

```
Overwriting mycalc.py
```

In [92]:
```
import mycalc

print(f"2 + 3 = {mycalc.add(2, 3)}")
print(f"5 - 2 = {mycalc.subtract(5, 2)}")
print(f"4 * 6 = {mycalc.multiply(4, 6)}")
print(f"10 / 2 = {mycalc.divide(10, 2)}")
print(f"7 / 0 = {mycalc.divide(7, 0)}")
```

```
2 + 3 = 5
5 - 2 = 3
4 * 6 = 24
10 / 2 = 5.0
7 / 0 = Lỗi: Không thể chia cho 0
```

In [93]:
```
# Bài 2: Viết một module geometry.py có hàm tính diện tích hình tròn, rồi sử dụn
%%writefile geometry.py
import math

def circle_area(radius):
  """Tính diện tích hình tròn."""
  if radius < 0:
    return "Lỗi: Bán kính không thể âm"
  return math.pi * (radius ** 2)
```

```
Writing geometry.py
```

In [94]:
```
# Tiếp tục Bài 2: Import và sử dụng module geometry.py
import geometry

# Ví dụ sử dụng hàm từ module geometry
print(f"Diện tích hình tròn bán kính 5 là: {geometry.circle_area(5)}")
print(f"Diện tích hình tròn bán kính -2 là: {geometry.circle_area(-2)}")
```

```
Diện tích hình tròn bán kính 5 là: 78.53981633974483
Diện tích hình tròn bán kính -2 là: Lỗi: Bán kính không thể âm
```

In [95]:
```
# Bài 3: Viết chương trình import statistics và tính mean, median, mode của list
import statistics

my_list = [1, 2, 3, 4, 4, 5, 6, 6, 6, 7]

print(f"Danh sách số: {my_list}")
print(f"Giá trị trung bình (mean): {statistics.mean(my_list)}")
print(f"Giá trị trung vị (median): {statistics.median(my_list)}")
print(f"Giá trị mode (phổ biến nhất): {statistics.mode(my_list)}")

# Ví dụ với danh sách có nhiều mode
my_list_multimode = [1, 1, 2, 2, 3]
try:
    print(f"Giá trị mode của {my_list_multimode}: {statistics.mode(my_list_multi
except statistics.StatisticsError as e:
    print(f"Lỗi khi tính mode cho {my_list_multimode}: {e}")
```

```
Danh sách số: [1, 2, 3, 4, 4, 5, 6, 6, 6, 7]
Giá trị trung bình (mean): 4.4
Giá trị trung vị (median): 4.5
Giá trị mode (phổ biến nhất): 6
Giá trị mode của [1, 1, 2, 2, 3]: 1
```

In [96]:
```
# Bài 4: Viết chương trình nhập vào một chuỗi, sử dụng built-in function để đếm
input_string = input("Nhập một chuỗi: ")

upper_count = sum(1 for char in input_string if char.isupper())
print(f"Số ký tự in hoa trong chuỗi '{input_string}' là: {upper_count}")
```

```
Nhập một chuỗi: JDIJVD
Số ký tự in hoa trong chuỗi 'JDIJVD' là: 6
```

In [97]:
```
# Bài 5: Viết chương trình sử dụng map() và lambda để tính lập phương của list [
my_list = [1, 2, 3, 4, 5]
cubed_list = list(map(lambda x: x**3, my_list))
print(f"Danh sách gốc: {my_list}")
print(f"Danh sách lập phương: {cubed_list}")
```

```
Danh sách gốc: [1, 2, 3, 4, 5]
Danh sách lập phương: [1, 8, 27, 64, 125]
```

In [98]:
```
# Bài 6: Tạo một module stringtools.py có hàm đếm số từ trong chuỗi. Gọi module
%%writefile stringtools.py
import re

def count_words(text):
  """Đếm số từ trong một chuỗi."""
  words = re.findall(r'\b\w+\b', text.lower())
  return len(words)
```

```
Writing stringtools.py
```

In [99]:
```
# Tiếp tục Bài 6: Gọi module stringtools.py từ file chính.
import stringtools

my_text = "Đây là một ví dụ về đếm từ trong chuỗi."
word_count = stringtools.count_words(my_text)
print(f"Số từ trong chuỗi '{my_text}' là: {word_count}")
```

```
Số từ trong chuỗi 'Đây là một ví dụ về đếm từ trong chuỗi.' là: 10
```

In [100…
```
# Bài 7: Viết chương trình sử dụng filter() để lọc ra các số nguyên tố trong lis

def is_prime(n):
  """Kiểm tra xem một số có phải là số nguyên tố hay không."""
  if n <= 1: return False
  if n <= 3: return True
  if n % 2 == 0 or n % 3 == 0: return False
  i = 5
  while i * i <= n:
    if n % i == 0 or n % (i + 2) == 0: return False
    i += 6
  return True

numbers = list(range(1, 51))
prime_numbers = list(filter(is_prime, numbers))
print(f"Các số nguyên tố trong khoảng từ 1 đến 50 là: {prime_numbers}")
```

```
Các số nguyên tố trong khoảng từ 1 đến 50 là: [2, 3, 5, 7, 11, 13, 17, 19, 23, 2
9, 31, 37, 41, 43, 47]
```

In [105…
```
# Bài 8: Viết chương trình dùng zip() để gộp danh sách tên sinh viên và điểm số,
student_names = ["Alice", "Nam", "Ha", "Nga"]
student_scores = [85, 92, 78, 95]

student_data = list(zip(student_names, student_scores))

print("Danh sách sinh viên và điểm số:")
for name, score in student_data:
  print(f"- {name}: {score}")
```

```
Danh sách sinh viên và điểm số:
- Alice: 85
- Nam: 92
- Ha: 78
- Nga: 95
```

In [106…
```
# Bài 9: Viết chương trình sử dụng built-in function sorted() để sắp xếp danh sá
people = [("Alice", 30), ("Nam", 25), ("Ha", 35), ("Nga", 28)]

sorted_by_age = sorted(people, key=lambda item: item[1])

print(f"Danh sách gốc: {people}")
print(f"Danh sách sắp xếp theo tuổi: {sorted_by_age}")
```

```
Danh sách gốc: [('Alice', 30), ('Nam', 25), ('Ha', 35), ('Nga', 28)]
Danh sách sắp xếp theo tuổi: [('Nam', 25), ('Nga', 28), ('Alice', 30), ('Ha', 3
5)]
```

In [108…
```
# Bài 10: Viết chương trình nhập danh sách số nguyên, dùng sum(), min(), max() đ
input_str = input("Nhập danh sách số nguyên cách nhau bởi dấu cách: ")
numbers_str = input_str.split()
numbers = [int(num) for num in numbers_str]
```

```
print(f"Danh sách số nguyên: {numbers}")
print(f"Tổng: {sum(numbers)}")
print(f"Min: {min(numbers)}")
print(f"Max: {max(numbers)}")
```

```
Nhập danh sách số nguyên cách nhau bởi dấu cách: 4 5 89 6 43
Danh sách số nguyên: [4, 5, 89, 6, 43]
Tổng: 147
Min: 4
Max: 89
```

In [109…

```
# Bài 11: Tạo một module matrix.py có hàm cộng hai ma trận. Import và kiểm tra k
%%writefile matrix.py
def add_matrices(matrix1, matrix2):
  """Cộng hai ma trận."""
  # Kiểm tra kích thước ma trận
  if len(matrix1) != len(matrix2) or len(matrix1[0]) != len(matrix2[0]):
    return "Lỗi: Hai ma trận phải cùng kích thước"

  rows = len(matrix1)
  cols = len(matrix1[0])
  result_matrix = [[0 for _ in range(cols)] for _ in range(rows)]

  for i in range(rows):
    for j in range(cols):
      result_matrix[i][j] = matrix1[i][j] + matrix2[i][j]

  return result_matrix
```

```
Writing matrix.py
```

In [110…

```
# Tiếp tục Bài 11: Import và kiểm tra kết quả module matrix.py
import matrix

matrix_a = [[1, 2], [3, 4]]
matrix_b = [[5, 6], [7, 8]]
matrix_c = [[1, 2, 3], [4, 5, 6]] # Ma trận khác kích thước

print(f"Ma trận A: {matrix_a}")
print(f"Ma trận B: {matrix_b}")
print(f"Kết quả A + B: {matrix.add_matrices(matrix_a, matrix_b)}")

print(f"\nMa trận C: {matrix_c}")
print(f"Kết quả A + C: {matrix.add_matrices(matrix_a, matrix_c)}")
```

```
Ma trận A: [[1, 2], [3, 4]]
Ma trận B: [[5, 6], [7, 8]]
Kết quả A + B: [[6, 8], [10, 12]]

Ma trận C: [[1, 2, 3], [4, 5, 6]]
Kết quả A + C: Lỗi: Hai ma trận phải cùng kích thước
```

In [111…

```
# Bài 12: Viết chương trình dùng module random để sinh ra 6 số ngẫu nhiên từ 1-4
import random

lottery_numbers = random.sample(range(1, 46), 6)
lottery_numbers.sort()
print(f"6 số xổ số ngẫu nhiên: {lottery_numbers}")
```

```
6 số xổ số ngẫu nhiên: [6, 17, 25, 26, 33, 43]
```

In [112…
```python
# Bài 13: Viết chương trình nhập một chuỗi, dùng built-in function để loại bỏ cá

input_string = input("Nhập một chuỗi: ")

unique_chars_set = "".join(set(input_string))
print(f"Chuỗi gốc: '{input_string}'")
print(f"Chuỗi sau khi loại bỏ trùng lặp (set): '{unique_chars_set}'")

unique_chars_ordered = ""
for char in input_string:
    if char not in unique_chars_ordered:
        unique_chars_ordered += char
print(f"Chuỗi sau khi loại bỏ trùng lặp (giữ thứ tự): '{unique_chars_ordered}'")
```

```
Nhập một chuỗi: hong yen
Chuỗi gốc: 'hong yen'
Chuỗi sau khi loại bỏ trùng lặp (set): 'nh ygoe'
Chuỗi sau khi loại bỏ trùng lặp (giữ thứ tự): 'hong ye'
```

In [113…
```python
# Bài 14: Viết chương trình sử dụng enumerate() để đánh số thứ tự các dòng đọc t
# Tạo một file văn bản giả định
%%writefile sample.txt
Đây là dòng đầu tiên.
Đây là dòng thứ hai.
Và đây là dòng cuối cùng.
```

```
Writing sample.txt
```

In [114…
```python
# Tiếp tục Bài 14: Sử dụng enumerate() để đọc và đánh số dòng từ file.
# Mở file để đọc
try:
  with open("sample.txt", "r") as file:
    # Dùng enumerate để duyệt qua các dòng và lấy số thứ tự
    for line_number, line in enumerate(file, 1): # Bắt đầu đếm từ 1
      print(f"{line_number}: {line.strip()}") # strip() để bỏ ký tự xuống dòng
except FileNotFoundError:
  print("Lỗi: Không tìm thấy tệp 'sample.txt'")
```

```
1: Đây là dòng đầu tiên.
2: Đây là dòng thứ hai.
3: Và đây là dòng cuối cùng.
```

In [116…
```python
# Bài 15: Viết chương trình nhập list số, dùng map() và abs() để lấy giá trị tuy
input_str = input("Nhập danh sách số cách nhau bởi dấu cách (có thể âm): ")
numbers_str = input_str.split()
numbers = [float(num) for num in numbers_str]

print(f"Danh sách gốc: {numbers}")

absolute_numbers = list(map(abs, numbers))
print(f"Danh sách giá trị tuyệt đối: {absolute_numbers}")
```

```
Nhập danh sách số cách nhau bởi dấu cách (có thể âm): 6 8 -9 -98 67 43
Danh sách gốc: [6.0, 8.0, -9.0, -98.0, 67.0, 43.0]
Danh sách giá trị tuyệt đối: [6.0, 8.0, 9.0, 98.0, 67.0, 43.0]
```

In [118…
```python
# Bài 16: Viết chương trình nhập list số, dùng filter() để lấy ra các số lớn hơn
input_str = input("Nhập danh sách số cách nhau bởi dấu cách: ")
numbers_str = input_str.split()
numbers = [float(num) for num in numbers_str]
```

```
print(f"Danh sách gốc: {numbers}")

numbers_greater_than_10 = list(filter(lambda x: x > 10, numbers))
print(f"Các số lớn hơn 10: {numbers_greater_than_10}")
```

```
Nhập danh sách số cách nhau bởi dấu cách: 6 76 4 2 5 6 7 8 76
Danh sách gốc: [6.0, 76.0, 4.0, 2.0, 5.0, 6.0, 7.0, 8.0, 76.0]
Các số lớn hơn 10: [76.0, 76.0]
```

In [120…

```
# Bài 17: Viết chương trình nhập list chuỗi, dùng sorted() để sắp xếp theo độ dà
input_str = input("Nhập danh sách chuỗi cách nhau bởi dấu cách: ")
string_list = input_str.split()

print(f"Danh sách chuỗi gốc: {string_list}")

sorted_by_length = sorted(string_list, key=len)
print(f"Danh sách sắp xếp theo độ dài: {sorted_by_length}")
```

```
Nhập danh sách chuỗi cách nhau bởi dấu cách: nguyen hong yen
Danh sách chuỗi gốc: ['nguyen', 'hong', 'yen']
Danh sách sắp xếp theo độ dài: ['yen', 'hong', 'nguyen']
```

In [121…

```
# Bài 18: Tạo một module converter.py có hàm đổi độ C sang độ F. Import và sử dụ
%%writefile converter.py
def celsius_to_fahrenheit(celsius):
  """Chuyển đổi độ C sang độ F."""
  fahrenheit = (celsius * 9/5) + 32
  return fahrenheit
```

```
Writing converter.py
```

In [122…

```
# Tiếp tục Bài 18: Import và sử dụng module converter.py
import converter

celsius_temp = 25
fahrenheit_temp = converter.celsius_to_fahrenheit(celsius_temp)
print(f"{celsius_temp} độ C bằng {fahrenheit_temp} độ F.")

celsius_temp = 0
fahrenheit_temp = converter.celsius_to_fahrenheit(celsius_temp)
print(f"{celsius_temp} độ C bằng {fahrenheit_temp} độ F.")
```

```
25 độ C bằng 77.0 độ F.
0 độ C bằng 32.0 độ F.
```

In [124…

```
# Bài 19: Viết chương trình nhập danh sách điểm, dùng built-in function để chuẩn
# Chuẩn hóa điểm = điểm / điểm_max
input_str = input("Nhập danh sách điểm cách nhau bởi dấu cách: ")
scores_str = input_str.split()
scores = [float(score) for score in scores_str]

print(f"Danh sách điểm gốc: {scores}")

if scores: # Kiểm tra danh sách không rỗng
  max_score = max(scores)
  if max_score > 0: # Tránh chia cho 0 nếu tất cả điểm là 0
    normalized_scores = [score / max_score for score in scores]
    print(f"Danh sách điểm chuẩn hóa: {normalized_scores}")
  else:
    print("Không thể chuẩn hóa: điểm tối đa là 0.")
```

```
else:
  print("Danh sách điểm rỗng.")
```

```
Nhập danh sách điểm cách nhau bởi dấu cách: 6 7.8 9.5 4.7 10
Danh sách điểm gốc: [6.0, 7.8, 9.5, 4.7, 10.0]
Danh sách điểm chuẩn hóa: [0.6, 0.78, 0.95, 0.47000000000000003, 1.0]
```

In [125…

```
# Bài 20: Viết chương trình sử dụng module datetime để in ra ngày tháng năm hiện
import datetime

now = datetime.datetime.now()

formatted_date = now.strftime("%d/%m/%Y")

print(f"Ngày tháng năm hiện tại: {formatted_date}")
```

```
Ngày tháng năm hiện tại: 18/10/2025
```