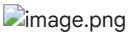
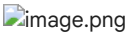
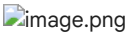




```
# hàm sprt trong thu viện math
import math
x = 16
can_bac_hai = math.sqrt(x)
print(can_bac_hai)
```

```
4.0
```



```
def hello_world():
    print("Hello World")
```

```
def greeting(name):
    print("Hello " + name)

greeting("Hùng")
```

```
Hello Hùng
```

```
def say_hello(name = "World"):
    print("Hello "+ name)

say_hello()
```

```
Hello World
```

```
def add(x, y):
    return x + y
```

```
def tinhtoan(x, y):
    tong = x + y
    hieu = x - y
    tich = x*y
    thuong = x/y if y != 0
    return tong, hieu, tich, thuong
```

*Hàm có tham số biến đổi(args): thêm dấu * vào trước tham số

Trong Python, *args cho phép một hàm nhận số lượng tham số không giới hạn (không cần biết trước có bao nhiêu tham số).

Các tham số truyền vào sẽ được gom lại thành một tuple.

args chỉ là tên thường dùng, bạn có thể đặt tên khác, nhưng phải có dấu * phía trước.

```
def sum_all(*args):
    return sum(args)
print(sum_all(1, 2, 3))
print(sum_all(2, 4, 6, 8))
```

Trong Python, khi định nghĩa hàm, bạn có thể dùng **kwargs để nhận số lượng tham số từ khóa (keyword arguments) không giới hạn.

kwargs là viết tắt của keyword arguments, tức là các tham số được truyền dưới dạng tên = giá trị.

Kết quả: bên trong hàm, kwargs sẽ là một dict chứa các cặp key–value.

```
# Ví du:
def print_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")
print_info(name = "Trung", age = 34)
print_info(name = "An", age = 20, monhoc = ["toan", "ly", "hoa"])
```

```
name: Trung
age: 34
name: An
age: 20
monhoc: ['toan', 'ly', 'hoa']
```

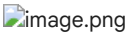
```
# Hàm tính tổng tiền từ nhiều sản phẩm
def tinh_tong(*args):
    return sum(args)

# Hàm thanh toán với nhiều tùy chọn
def thanh_toan(*args, **kwargs):
    tong = tinh_tong(*args)

    # Xử lý kwargs (tham số từ khóa)
    giam_gia = kwargs.get("giam_gia", 0)     # %
    phi_ship = kwargs.get("phi_ship", 0)
    thue = kwargs.get("thue", 0)             # %

    tong -= tong * giam_gia / 100
    tong += phi_ship
    tong += tong * thue / 100

    return tong


# ---- Ví dụ sử dụng ----
print("Tổng tiền hàng:", tinh_tong(100, 200, 300))

tong_thanh_toan = thanh_toan(100, 200, 300, giam_gia=10, phi_ship =3 0, thue = 5)
print("Khách phải trả:", tong_thanh_toan)
```

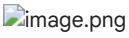
```
def giai_phuong_trinh_bat_nhat(he_so_b, he_so_c):
    if he_so_b == 0:
        if he_so_c == 0:
            return "Phương trình Vô số nghiệm"
        else:
            return "Phương trình vô nghiệm"
    else:
        return -he_so_c / he_so_b

giai_phuong_trinh_bat_nhat(7, 4)
```

```
# đây dãy số tạo sẵn trong bài lab dãy số này nhập từ bàn phím
day_so = []
while True:
    print("Nhập phần tử cho dãy số:")
    phan_tu = input("nhập phần từ:")
    if phan_tu.lower() == "q":
         print("kết thúc việc nhập")
         break
    else:
        try:
            so = int(phan_tu)
            day_so.append(so)
        except ValueError:
            print("Vui lòng nhậ đúng giá trị số hoặc q để thoát")
print(day_so)
# khai báo biến tong và dem ban đầu =0
tong = 0
dem = 0

# duyệt dãy số dùng for phan_tu in day so:
for phan_tu in day_so:
    # kiểm tra phần từ trong dãy số chia hết cho 3 hay không
    if phan_tu % 3 == 0:
```

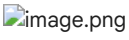
```
        tong = tong + phan_tu
        dem = dem + 1
print(f"  có {dem} chia hết cho 3 trong day số có tổng {tong}")
```

lambda trong python





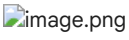
```
# Ví dụ:
# định nghĩa hàm thông thường
def miles_to_km(miles):
    return miles * 1.6
# dùng lambda
kms = lambda miles :miles * 1.6
```



```
# ví du:
# định nghĩa hàm thông thường để kiểm tra 1 số chẵn

def kiem_tra_so_chan(so_can_kiem_tra):
    if so_can_kiem_tra % 2 ==0:
        return True
    else:
        return False
# dùng lambda
ket_qua = lambda so_can_kiem_tra: True if so_can_kiem_tra % 2 ==0 else False
print(ket_qua(15))
```

```
False
```

Map & Filter trong python



```
# ví dụ về map() tính bình phuong các phần từ treong list
day_so = [1, 3, 4, 6, 9]

# dùng hàm tính bình phương
def tinh_binh_phuong(x):
    return x * x
day_so_binh_phuong = map(tinh_binh_phuong, day_so)
# In kết quả: Vì kết quả trả về là map object, nên chúng ta cần biến đổi sang list
print(list(day_so_binh_phuong))
```

```
[1, 9, 16, 36, 81]
```

```
# kết hợp map() và lambda
day_so = [1, 3, 4, 6, 9]
day_so_binh_phuong = map(lambda x: x * x, day_so)
# In kết quả: Vì kết quả trả về là map object, nên chúng ta cần biến đổi sang list
print(list(day_so_binh_phuong))
```

```
[1, 9, 16, 36, 81]
```

```
# ví dụ về filter()
day_so = [1, 3, 4, 5, 6, 8, 9, 11, 23, 49]
def so_chan(x):
    if x % 2 == 0:
        return True
day_so_chan = filter(so_chan, day_so)
# In kết quả: Vì kết quả trả về là filter object, nên chúng ta cần biến đổi sang list
print(list(day_so_chan))
```

```
[4, 6, 8]
```

```
# Ví dụ 2: Lọc các phần tử trong my_list sang new_list nếu là phần tử đó
# là chuỗi
danh_sach = ["hello", "world", 1, 4, 9, "good bye", 4.5]
def la_chuoi(x):
    if type(x) == str:
        return True
danh_sach_chuoi = filter(la_chuoi, danh_sach)
print(list(danh_sach_chuoi))
```

```
['hello', 'world', 'good bye']
```

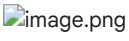
```
# kết hợp filter và lambda
# ví dụ về filter()
day_so = [1, 3, 4, 5, 6, 8, 9, 11, 23, 49]

day_so_chan = filter(lambda x: x % 2 == 0, day_so)
# In kết quả: Vì kết quả trả về là filter object, nên chúng ta cần biến đổi sang list
print(list(day_so_chan))
```

```
[4, 6, 8]
```

```
# Ví dụ 2: Lọc các phần tử trong my_list sang new_list nếu là phần tử đó
# là chuỗi
danh_sach = ["hello", "world", 1, 4, 9, "good bye", 4.5]

danh_sach_chuoi = filter(lambda x: type(x) == str, danh_sach)
print(list(danh_sach_chuoi))
```

```
['hello', 'world', 'good bye']
```

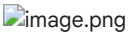
Global & local Variables



```
# Biến global (toàn cục)
x = 10

def my_function():
    # Biến local (cục bộ)
    y = 5
    print("Bên trong hàm, x =", x)   # có thể truy cập biến global
    print("Bên trong hàm, y =", y)   # chỉ tồn tại trong hàm

my_function()

print("Bên ngoài hàm, x =", x)   # vẫn dùng được
print("Bên ngoài hàm, y =", y)  #  lỗi, vì y là biến local trong hàm
```

```
Bên trong hàm, x = 10
Bên trong hàm, y = 5
Bên ngoài hàm, x = 10
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 13
     10 my_function()
     12 print("Bên ngoài hàm, x =", x)   # vẫn dùng được
---> 13 print("Bên ngoài hàm, y =", y)  #  lỗi, vì y là biến local trong hàm

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

Try Except



```
# Ví Du:
# nhập dãy số và dừng lại bất cứ khi nào :
day_so = []
while True:
    try:
        so = input("nhap số")
        if so.lower() != "q":
            phan_tu = int(so)
            day_so.append(so)
        else:
            print("Thoat")
            break
    except ValueError:
        print("Vui lòng nhập số hoặc phím q để thoát")
    finally:
        print("Đã nhập xong day số")
print(day_so)
```

```
Vui lòng nhập số hoặc phím q để thoát
Đã nhập xong day số
Đã nhập xong day số
Đã nhập xong day số
Đã nhập xong day số
Đã nhập xong day số
Đã nhập xong day số
Đã nhập xong day số
Đã nhập xong day số
Đã nhập xong day số
Thoat
Đã nhập xong day số
['1', '3', '4', '5', '6', '7', '8', '675']
```