

# Ask Claude

---

## Me

import文で取り込まれると、関数や変数のみが定義されたファイルであっても、オブジェクトになると考えればよいでしょうか。

## Claude

はい、その通りです！ Pythonファイル（.py）は、関数や変数のみが定義されたファイルであっても、importされると**モジュールオブジェクト**になります。

### 例：シンプルなモジュール

```
# mymodule.py
# クラスは一切定義していない
greeting = "Hello"
number = 42

def say_hello(name):
    return f"{greeting}, {name}!"

def add(a, b):
    return a + b
```

```
# mymodule_test.py
# 使用時
import mymodule

print(type(mymodule))          # <class 'module'> ← オブジェクト！
print(mymodule.greeting)      # "Hello"
print(mymodule.say_hello("Python")) # "Hello, Python!"
```

### 空のファイルでもモジュールオブジェクトに

```
# empty.py（空のファイル）
# 何も書いていない
```

```
# empty_test.py
# 使用時
import empty
print(type(empty)) # <class 'module'> ← これもオブジェクト！
```

## 自作classをインポートする場合

```
# dog.py
class Dog:
    def __init__(self, name, age):
        self.name = name # データ（属性）
        self.age = age # データ（属性）

    def bark(self): # メソッド（bark：吠える）
        return f"{self.name}が吠えています：ワンワン！"

    def get_info(self): # メソッド（info：information,情報）
        return f"{self.name}は{self.age}歳です"
```

```
# dog_test.py
from dog import Dog

# これはダメ
# print(Dog.bark()) # エラー：selfに渡すインスタンスがない
# print(Dog.name)   # エラー：クラスにはname属性がない

# 正しい方法
# インスタンスを作成（初期化）
my_dog = Dog("ポチ", 3)

# インスタンスのメソッドを呼び出す
print(my_dog.bark()) # ポチが吠えています：ワンワン！
print(my_dog.get_info()) # ポチは3歳です

# インスタンスの属性にアクセス
print(my_dog.name) # ポチ
print(my_dog.age) # 3
```

## まとめ

- classでなくても `.pyファイル` を、importすると**モジュールオブジェクト**になる
- よく知られているモジュールは、定番の使い方
- 自作classの時は、**インスタンス化（初期化）**を忘れないこと！

## importの考え方

データ名



```
from ファイル名 import 関数名/フィールド名/クラス名
```

## 複数をインポートする書き方

```
from mymodule import func1, func2, var1    # カンマ区切り
```

```
from mymodule import (func1, func2, var1)  # タプル形式（括弧あり）
```

```
from mymodule import (  
    very_long_function_name,  
    another_long_function_name,  
    some_constant_value,  
    MyClass  
)
```

END