Fresher **sự khác nhau giữa list, tuple**

- List có thể thay đổi, có thứ tự, có thể trùng giá trị
- Tuple không thể thay đổi, có thứ tự, có thể trùng giá trị
- Set có thể thay đổi(thêm - xóa), không có thứ tự và không trùng lặp giá trị, phần tử trong Set là bất biến

Fresher **List comprehension là gì**

- cú pháp ngắn gọn tạo ra list từ 1 list có sẵn
- [expression for item in iterable if condition]

Fresher **virtualenv là gì**

- Tạo môi trường ảo tách biệt các gói thư viện cài đặt cho 1 dự án mà không ảnh hưởng tới các dự án khác, tránh xung đột
- Quản lí version của từng thư viện

pip install virtualenv

virtualenv myenv

myenv\Scripts\activate

deactivate

pip freeze > requirements.txt

pip install -r requirements.txt

Fresher **Sử dụng thư viện nào để viết unit test**

Trong dự án thực tế, cần viết mỗi tệp test cho mỗi file mã nguồn

1. **Pytest**

- Hỗ trợ assertion, tích hợp tốt các công cụ CI/CD
- Mocking với pytest-mock

2. **Unittest**

- Tạo class kế thừa từ unittest.TestCase
- Định nghĩa các phương thức bắt đầu bằng test_
- Sử dụng các phương thức kiểm tra như assertEqual(), assertTrue(), assertFalse().

3. **Nose2, doctest**

Fresher Cách debug với python

- Sử dụng print() thêm vào các vị trí của work flow tìm lỗi
- Sử dụng module logging: ghi lại log theo thời gian thực để giám sát, chia log thành các loại: DEBUG, INFO, WARNING, ERROR, CRITICAL

  import logging
  logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')
- Sử dụng module assert: dùng trong quá trình phát triển
  - Cú pháp: assert CONDITION, "message"
- Sử dụng IDE debug: Step over – chạy thêm 1 bước, ko vào hàm; Step into – vào hàm; Step out – thoát hàm; Continue – chạy tiếp tới checkpoint
  - **Variables Panel:** xem giá trị các biến trong phạm vi hiện tại
  - **Watch variables:** thêm biến cụ thể để theo dõi toàn bộ quá trình thay đổi
  - **Evaluate expression:** thêm mã để chạy ngay lập tức tại breakpoint mà không làm thay đổi mã của CT
  - Thay đổi ngay lập tức giá trị của biến trong cửa sổ panel
  - Đặt breakpoint có điều kiện, nếu thoải mãn thì breakpoint mới hoạt động
- Sử dụng module **pdb** trong python: **import pdb; pdb.set_trace()**
  Khi chương trình gặp dòng trên sẽ dừng lại chuyển vào chế độ debug và có thể nhập vào command line:

  n (next): Thực thi dòng hiện tại, không bước vào hàm, mà dừng ở dòng tiếp theo trong cùng hàm hiện tại.

  s (step): Thực thi dòng hiện tại, bước vào hàm nếu có lời gọi hàm.

  c (continue): Tiếp tục chạy cho đến khi gặp breakpoint tiếp theo.

  l (look): xem mã xung quanh vị trí checkpoint

  p variable_name: In giá trị của biến.

  q (quit): Thoát khỏi chế độ debug và dừng chương trình.

  break <số dòng> : thêm checkpoint khi debug với pdb

  clear <số thứ tự breakpoint> : xóa breakpoint

Fresher __init__, __str__ có chức năng gì

- **Là các magic methods/dunder methods**
- __init__ : constructor có var đầu tiền là self. **def __init__(self, name, age):**
- __str__ : định nghĩa cách chuyển đối tượng thành String. **def __str__(self):**

Phương thức	Chức năng
`__init__`	Hàm khởi tạo đối tượng
`__del__`	Hàm hủy đối tượng (destructor)
`__str__`	Định nghĩa cách đối tượng được in ra (chuỗi đại diện)
`__repr__`	Định nghĩa cách đối tượng được đại diện (dùng cho debugging)
`__eq__`, `__ne__`, ...	Các phương thức so sánh (==, !=, <, >, ...)
`__add__`, `__sub__`, ...	Các phương thức toán tử (+, -, *, /, ...)
`__call__`	Cho phép đối tượng được gọi như một hàm
`__getattr__`, `__setattr__`, `__delattr__`	Các phương thức truy cập thuộc tính
`__enter__`, `__exit__`	Quản lý context (dùng trong `with` statement)

Fresher Dùng *args, **kwargs như thế nào

*args: Dùng để truyền một số lượng đối số không xác định và chúng sẽ được gom thành một tuple.

**kwargs: Dùng để truyền một số lượng đối số từ khóa không xác định và chúng sẽ được gom thành một dictionary.

Bạn có thể sử dụng cả *args và **kwargs trong một hàm, nhưng *args phải luôn đứng trước **kwargs.

Fresher Biến toàn cục và cục bộ khác nhau ra sao

- Biến toàn cục không được khai báo trong bất kì hàm nào, có thể truy cập mọi nơi trong CT, sử dụng từ khóa global nếu muốn thay đổi giá trị của nó trong 1 hàm
- Biến cục bộ là biến được sử dụng trong 1 khối mã nào đó (hàm, vòng lặp,…)

Fresher Lambda function là gì?

- Là hàm vô danh với cú pháp ngắn gọn, hàm nhỏ, sử dụng 1 lần
- **lambda arguments: expression**

```
# Lambda function cộng hai số
add = lambda a, b: a + b
```

```python
numbers = [1, 2, 3, 4]
squared = list(map(lambda x: x ** 2, numbers))
print(squared)  # In ra [1, 4, 9, 16]
```

```python
numbers = [1, 2, 3, 4, 5]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)  # In ra [2, 4]
```

```python
# Danh sách các tuple, sắp xếp theo phần tử thứ hai của mỗi tuple
pairs = [(1, 2), (3, 1), (5, 4)]
sorted_pairs = sorted(pairs, key=lambda x: x[1])
print(sorted_pairs)  # In ra [(3, 1), (1, 2), (5, 4)]
```

Cách dùng sorted()

Fresher GIL là gì? Có ảnh hưởng gì khi dùng đa luồng?

- GIL (Global Interpreter Lock): cơ chế quản lí quyền truy cập vào bộ nhớ khi chạy đa luồng trong python
- GIL chỉ cho phép 1 luồng thực hiện mã python bytecode tại 1 thời điểm. Nếu có nhiều luồng CPU-bound thì chúng lần lượt chiếm quyền CPU thay vì sử dụng các nhân CPU khác nhau. Còn đối với các luồng I/O-bound(đọc ghi tệp, truy vấn csdl) thì Python giải phòng GIL cho phép các luồng khác thực thi khi chờ đợi 1 luồng
- Cách giải quyết GIL là:
    - Sử dụng **multiprocessing** thay vì **threading**: multiprocessing chạy trên các process độc lập, các process chạy song song trên CPU core khác nhau và có không gian bộ nhớ riêng không bị ảnh hưởng bởi GIL
    - Sử dụng thư viện bên ngoài như Numpy Cython Pypy

Fresher Decorator là gì? Cách viết một decorator?

- Decorator là một kiểu hàm đặc biệt, cho phép bạn thêm chức năng mới vào các hàm hoặc phương thức mà không cần thay đổi trực tiếp mã của chúng
- Decorator là một hàm nhận vào một hàm khác và trả về một hàm mới
- Decorator giúp bạn tái sử dụng mã để thêm chức năng vào nhiều hàm mà không cần phải thay đổi từng hàm.
- Decorators giúp bạn tách các chức năng phụ ra khỏi hàm chính và dễ dàng duy trì mã. (hạn như logging, kiểm tra quyền truy cập, đo lường hiệu suất)
- Functools.wraps giúp bảo toàn các thuộc tính của hàm gốc, như tên và docstring