Ficha de ayuda: Pytest

La comprobación Pytest es una función que comienza con el **prefijo** `test`. Esta distingue entre mayúsculas y minúsculas, por ejemplo, `test1` funcionará bien, pero `Test1` no.

Instalación de Pytest

Hay dos maneras de instalar Pytest.

1️⃣ Mediante el comando "pip" en la consola

Abre la consola. Escribe `pip install pytest`.

En este caso, `pip` es un administrador de paquetes integrado en Python. Esto significa que es un programa que te permite instalar varios paquetes, ya está integrado en Python.

> 📎 Si el comando `pip` no funciona, intenta usar `pip3` en su lugar.

2️⃣ Mediante la pestaña "Python Packages" (Paquetes de Python)

1. Abre tu proyecto, encuentra el panel inferior y haz clic en la pestaña "Python Packages".
2. Escribe "Pytest" en el campo de búsqueda.
3. Selecciona el paquete "Pytest" y haz clic en "Install" (Instalar).

Ejecución de pruebas

Puedes ejecutar tu prueba desde la consola o desde la interfaz de PyCharm.

1️⃣ Ejecución de pruebas desde la terminal de PyCharm

Abre la pestaña "Terminal" en la barra de herramientas auxiliares. De forma predeterminada, el directorio de trabajo en esta terminal es la carpeta que contiene tu proyecto actual.

Ejecuta todas las pruebas del proyecto:

```
pytest
```

Luego ejecuta las pruebas desde el archivo `calc_test.py`:

```
pytest calc_test.py
```

2️⃣ Ejecución de pruebas desde la interfaz de PyCharm

Ejecuta todas las pruebas del proyecto:

1. En el menú "Run", selecciona "Edit Configurations" (Editar configuraciones).

2. Haz clic en "+ (Add New Configuration)" (Agregar nueva configuración) en una ventana nueva.
3. Selecciona "Python tests → pytest" (Pruebas de Python → Pytest) en la lista de configuraciones.
4. Aparecerá una ventana nueva. Selecciona "Custom" (Personalizar) en la línea "Target" (Objetivo).
5. Haz clic en "Apply" (Aplicar) y luego en "OK" (Aceptar).
6. Ahora haz clic en el ícono de la flecha verde para iniciar la configuración y observa los resultados.

Ejecuta las pruebas en un archivo determinado:

1. En el menú "Run", selecciona "Edit Configurations" (Editar configuraciones).
2. Haz clic en "+ (Add New Configuration)" (Agregar nueva configuración) en una ventana nueva.
3. Selecciona "Python tests → pytest" (Pruebas de Python → Pytest) en la lista de configuraciones.
4. Aparecerá una ventana nueva. El "Script path" está seleccionado en la línea "Target" de forma predeterminada. No cambies nada aquí.
5. Selecciona el archivo con tus pruebas en el campo debajo de "Target".
6. Haz clic en "Apply" (Aplicar) y luego en "OK" (Aceptar).
7. Ahora haz clic en el ícono de la flecha verde para iniciar la configuración y observa los resultados:

Inicia una prueba en el archivo:

Después de instalar Pytest, verás una flecha verde junto a las pruebas que se utiliza para ejecutarlas. Haz clic en ella y observa el resultado.

La librería Requests de Python

Puedes instalar la librería requests mediante la consola o la pestaña "Python Packages" (Paquetes de Python).

Solicitud GET

1. Crea el archivo `configuration.py` en tus proyectos. La URL y la ruta de la documentación se almacenarán allí.

```
URL_SERVICE = "copia la URL sin la barra inclinada al final"
DOC_PATH = "/docs/"
```

2. Crea otro archivo y nómbralo `sender_stand_request.py`. Importa a este el archivo `configuration` y el paquete `requests` utilizando la palabra clave `import`. Esta se indica arriba. Aquí está tu código:

```
import configuration
import requests
```

3. Para enviar la solicitud mediante la librería Requests de Python, escribe el nombre de la librería y utiliza la función `get()`. Cuando se llama a la función, esta recibe una ruta completa a la documentación. Además, se le pueden pasar los encabezados y parámetros de la solicitud.

Propósito del atributo	Atributo	Ejemplo de uso
Diccionario de los encabezados de solicitud	headers	`headers={"Content-Type": "application/json"}`
Diccionario de los parámetros de solicitud	params	`params={"count":2}`

```
def get_docs():
    return requests.get(configuration.URL_SERVICE + configuration.DOC_PATH)


response = get_docs();
print(response.status_code)
```

Junto con el código de respuesta, puedes recibir otros atributos. A continuación te mostramos un ejemplo:

Nombre del atributo	Propósito
headers	Encabezados de respuesta
status_code	Código de respuesta
ok	Devuelve "True" si el código es 2xx o 3xx
url	Dirección de solicitud
request	Método de solicitud
text	Cuerpo de respuesta en formato de texto
json()	Pasa los datos de respuesta al diccionario, de lo contrario, se devolverá un error

Solicitud POST

1. Agrega la ruta al archivo `configuration.py`. A continuación te mostramos un ejemplo:

```
CREATE_USER_PATH = "/api/v1/users/"
```

2. Crea un archivo separado `data.py` para los datos de la solicitud. Este contendrá el cuerpo de solicitud y los encabezados.

```
headers = {
    "Content-Type": "application/json"
}

user_body = {
    "firstName": "Andrea",
    "phone": "+11234567890",
    "address": "123 Elm Street, Hilltop"
}
```

3. Importa el archivo `data` al archivo `sender_stand_request.py`. Así se ven los datos importados:

```
import configuration
import requests
import data
```

4. Para enviar la solicitud mediante la librería Requests de Python, escribe el nombre de la librería y llama a la función `POST()`. Al llamar a esta función, pasa el cuerpo completo de solicitud y la ruta a la misma. También puedes incluir los encabezados de solicitud.

Propósito del atributo	Atributo	Ejemplo de uso
Diccionario del cuerpo de solicitud en JSON	json	`json={ "firstName": "Andrea", "phone": "+11234567890", "address": "123 Elm Street, Hilltop" }`
Diccionario de los encabezados de solicitud	encabezados	`headers={"Content-Type": "application/json"}`

```
def post_new_user(body):
    return requests.post(configuration.URL_SERVICE + configuration.CREATE_USER_PATH,  # inserta la dirección URL completa
                         json=body,  # inserta el cuerpo de solicitud
                         headers=data.headers)  # inserta los encabezados


response = post_new_user(data.user_body);
print(response.status_code)
```

Reglas básicas para escribir autotests

Las autotests perfectas requieren una estructura perfecta. Hay tres reglas principales:

- Una prueba, una comprobación.
- Los datos deben ser independientes.
- Las pruebas deben ser independientes.