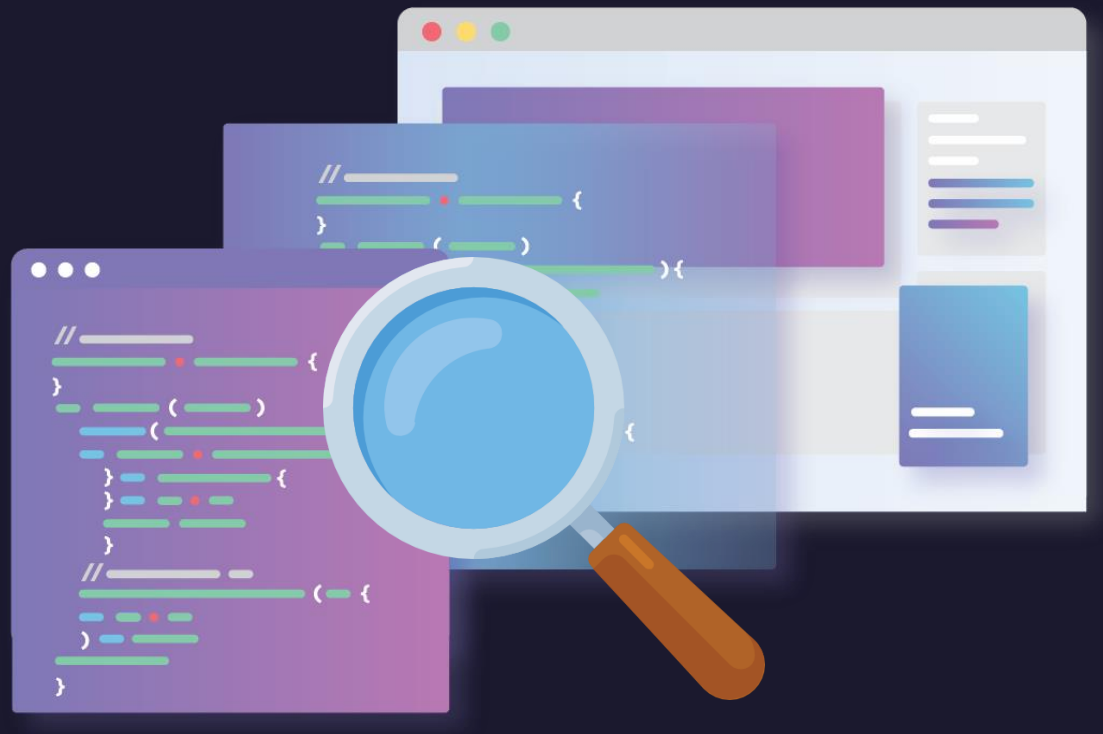
Introducción al Unit Tesing

ISIS 1221 – Sección 10

¿Qué es Unit Testing?

El *unit testing* o pruebas unitarias es una forma de comprobar que nuestros bloques de código (funciones) se ejecuten según su comportamiento esperado.



Características de un Unit Test

- **Automatiza el código:** Permiten probar el código sin intervención manual.
- **Completo:** Cubre la mayor cantidad de código.
- **Reutilizable:** Deben poder correr cada vez que se hacen cambios.
- **Independiente:** La ejecución de una no debe afectar a las demás pruebas.
- **Profesional:** Deben seguir estándares de código limpio y documentación.

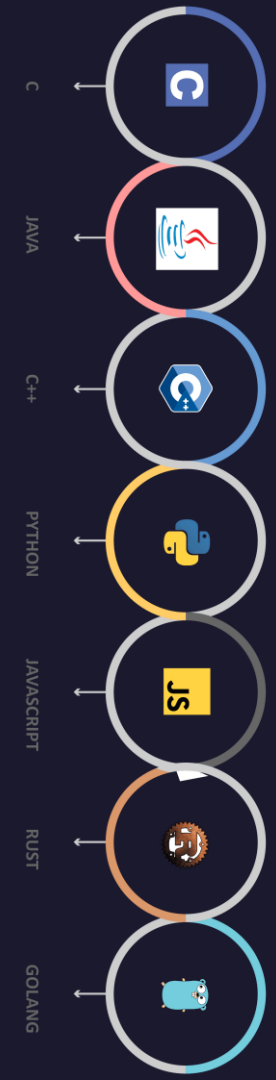
Ventajas

- **Fomentan el cambio:** Permite realizar cambios para agregar funcionalidades, optimizar el código, etc. Sin afectar el funcionamiento.
- **Simplifica la integración:** Facilita la integración ya que permite saber si el código funciona correctamente.
- **Documenta el código:** Documentan el código ya que desde las pruebas se puede identificar como se usa el código.
- **Separación de la interfaz y la implementación:** No se ven afectados por cambios en la interfaz.
- **Los errores están más acotados y son más fáciles de localizar:** Es posible identificar que partes del código no funcionan.

Limitaciones

No son infalibles :(
No pueden identificar errores de diseño.

Herramientas



- Typemock
- JUnit
- TestNG
- Jtiger
- SimpleTest
- PHPUnit
- CPPUnit
- Nunit
- FoxUnit
- MOQ
- Qunit
- Libunittest
- CUnit:
- PyUnit
- QtTest
- utPLSQL

¿Cómo funcionan los Unit Test?

Definir el test

Crear escenarios de prueba

Ejecutar la función a probar

Confirmar que el valor esperado sea el que devuelve la función

```
def test_buscar_amigo_por_nombre  (self):

amigo1 = {'nombre': 'Pedro Sánchez', 'fecha_de_nacimiento': 19960117, 'signo_zodiacal': 'CAPRICORNIO',
    'genero': 'M', 'genero_musical_favorito': 'pop', 'genero_literaro_favorito': 'drama',
    'numero_de_likes': 100, 'numero_de_publicaciones': 20, 'amigos_en_comun': 5, 'bloqueado': False}

 amigo2 = {'nombre': 'Luna Ariza', 'fecha_de_nacimiento': 20030510, 'signo_zodiacal': 'TAURO',
    'genero': 'O', 'genero_musical_favorito': 'pop', 'genero_literaro_favorito': 'drama',
    'numero_de_likes': 80, 'numero_de_publicaciones': 10, 'amigos_en_comun': 3, 'bloqueado': False}

amigo3 = {'nombre': 'Paula Hernández', 'fecha_de_nacimiento': 20011108, 'signo_zodiacal': 'ESCORPIO',
    'genero': 'F', 'genero_musical_favorito': 'salsa', 'genero_literaro_favorito': 'ciencia ficción',
    'numero_de_likes': 107, 'numero_de_publicaciones': 50, 'amigos_en_comun': 2, 'bloqueado': False}

amigo4 = {'nombre': 'Tobías Fuentes', 'fecha_de_nacimiento': 19990327, 'signo_zodiacal': 'ARIES',
    'genero': 'M', 'genero_musical_favorito': 'rap', 'genero_literaro_favorito': 'lírico',
    'numero_de_likes': 20, 'numero_de_publicaciones': 1, 'amigos_en_comun': 0, 'bloqueado': True}


   encontrado = cb.buscar_amigo_por_nombre("Pedro Sánchez", amigo1, amigo2, amigo3, amigo4)


   self.assertEqual(amigo1, encontrado, "La función no encontró el sector buscado")
```

Busca un sector por nombre
En este caso busca el sector con nombre “sector a” que es el s1

Estructura de un Unit Test en Python

Importamos la librería de test de Python

Definimos la clase en la que agrupamos todos nuestros test

Ejecutamos todos los test al correr el archivo

```
import unittest

class MyTestCase(unittest.TestCase):
    def test_something(self):
        self.assertEqual(True, False)

    #Aqui colocas tus tests

if __name__ == '__main__':
    unittest.main()
```

Establecemos nuestras pruebas Unitarias

¿Cómo ejecutar los test del Proyecto de nivel 2?

1

Descargar el archivo

Test_calculadora_de_velocidades.py

2

Incluir el archivo en la carpeta del proyecto

3

Abrir el archivo en Spyder

y correr el archivo en Spyder

Analizar resultados

4

```
################### Test usuario_mas_compatibilidad ###################################################
Verificando la busqueda del amigo en la posicion 1 ---------------------------------------- ✓
Verificando la busqueda del amigo en la posicion 2 ---------------------------------------- ✓
Verificando la busqueda del amigo en la posicion 3 ---------------------------------------- ✓
Verificando la busqueda del amigo en la posicion 4 ---------------------------------------- ✓

✓Test:
[                                                                                ] 100.0% done

----------------------------------------------------------------------
Ran 10 tests in 0.046s

OK
```

Si pasas todos los test significa que tus funciones se ejecutaron según lo esperado :D

```
✗Test:
[                                                                                ] 91.6% done
======================================================================
FAIL: test_asignar_signo_zodiacal (__main__.TestCupiBook)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\juand\Git\ISIS1221_IP\202210_10\Nivel 2\test_cupibook.py", line 308, in
test_asignar_signo_zodiacal
    self.assertEqual(caso_aries2, "ARIES", "La función no devolvió o encontró el erroneo")
AssertionError: 'GÉMINIS' != 'ARIES'
- GÉMINIS
+ ARIES
 : La función no devolvió o encontró el erroneo
```

Apóyate de la información extra.

Si no pasaste alguno, revisa cual fue el que no pasaste y corrige el código.

```
Verificando la asignación de aries como signo zodiacal para la fecha 20000321 ------------ ✓
Verificando la asignación de aries como signo zodiacal para la fecha 20000620 ------------ ✗
```