Testování, Unit testování a dokumentace zdrojového kódu

Testování

Testování softwaru je součástí vývoje softwaru. Jedná se o technický výzkum produktu a odhalování chyb programech. Účelem testování je odhalení chyb, zvýšení spolehlivosti a důvěryhodnosti a poskytnutí zpětné vazby. Jedná se také o simulaci běhu. Podle průzkumů oprava chyby v produkci stojí 10-1000× více než oprava během vývoje.

Testovací cyklus

1. Plánování testů
2. Analýza a příprava testů
3. Vykonání testů
4. Reportování výsledků

Každá část má vlastní reporting

FURPS (Functionality, Usability, Reliability, Performance, Supportability)
Furps je model popisující klíčové aspekty kvality softwaru:

1. Functionality - zaměřuje se na správně chování funkcí systému podle sepecifikace
2. Usability - uživatelské rozhraní a user experience
3. Reliability - stabilita systému, odolnost proti chybám a zotavení
4. Performance - rychlost, odezva a využití zdrojů
5. Podpora - instalace, konfigurace, aktualizace

další aspekty - Přenositelnost, Kompatibilita, Bezpečnost atd….

Dělení testů podle metod
White box - tester zná interní implementaci kódu, testuje konkrétní části kódu a jejich fungování, využívá znalost algoritmů a logiky programu, typicky programátor
Black box - tester nezná interní implementaci, testuje pouze vstupy a výstupy - simuluje pohled uživatele, typicky tester bez znalosti kódu
Gray box - kombinace obou přístupů, tester zná částečně strukturu ale ne detaily implementace

Realizace testů
Manuální - manuální proklikání/prozkoušení funkcí
Automatizované - napsané testy očekávající výsledek po akci
Exploratory -

Unit testování

Unit testování je proces testování nejmenších částí kódu - jednotlivých funkcí nebo metod. Testy by neměly ověřovat stav interakcí příliš mnoha nesouvisejících součástí a neměly by se překrývat. V Pythonu k tomu slouží modul unittest, který poskytuje framework pro organizaci a spouštění testů.

```
import unittest

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

class Unittest(unittest.TestCase):
   def test_units(self):
       self.assertEqual(add(1, 2),3)
```

Method	Checks that
`assertEqual(a, b)`	`a == b`
`assertNotEqual(a, b)`	`a != b`
`assertTrue(x)`	`bool(x) is True`
`assertFalse(x)`	`bool(x) is False`
`assertIs(a, b)`	`a is b`
`assertIsNot(a, b)`	`a is not b`
`assertIsNone(x)`	`x is None`
`assertIsNotNone(x)`	`x is not None`
`assertIn(a, b)`	`a in b`
`assertNotIn(a, b)`	`a not in b`
`assertIsInstance(a, b)`	`isinstance(a, b)`
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`

Dokumentace zdrojového kódu
Dokumentace zdrojového kódu je textový popis, který má za úkol vysvětlovat, jak kód funguje a jak je používán. Hezky vytvořená dokumentace nám značně usnadňuje údržbu, rozšiřování a používání kódu.

Typy dokumentace
Vnější - README, tutoriály a návody…
existuje samostatně mimo kód
Vnitřní - komentáře, docstringy, popisné názvy proměn
existuje v kódu

Prostě asi glazovat mandíka za to jak je to hrozně důelžitý, nějaký vlastní zkušenosti