

# Python: Testowanie oprogramowania

Jerzy Grynczewski

# Plan na dzisiaj

- ❖ Wstęp

# Plan na dzisiaj

- ❖ Wstęp
- ❖ Piszemy pierwsze testy

# Plan na dzisiaj

- ❖ Wstęp
- ❖ Piszemy pierwsze testy
- ❖ Biblioteka pytest (funkcje testowe)

# Plan na dzisiaj

- ❖ Wstęp
- ❖ Piszemy pierwsze testy
- ❖ Biblioteka pytest (funkcje testowe)
- ❖ Biblioteka unittest (klasy testowe)

# Plan na dzisiaj

- ❖ Wstęp
- ❖ Piszemy pierwsze testy
- ❖ Biblioteka pytest (funkcje testowe)
- ❖ Biblioteka unittest (klasy testowe)
- ❖ Zagadnienia zaawansowane

# Po co testować ?



Ariane 5 – int overflow, który wysadził rakietę w powietrzu

# Ariane 5

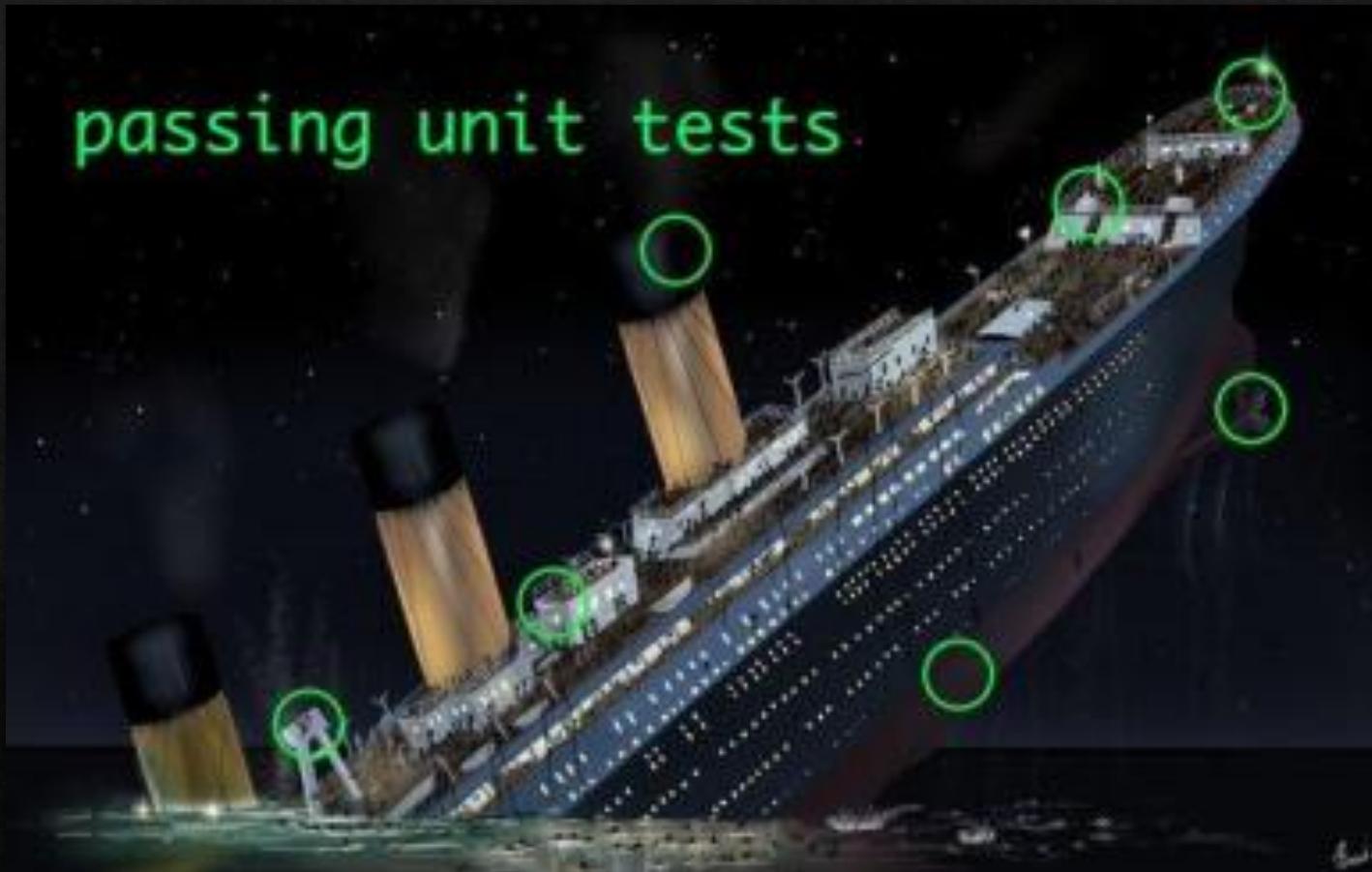
## Why visibility matters—the Ariane 5 crash

- Velocity was represented as a 64-bit float
- A conversion into a 16-bit signed integer caused an overflow
- The current velocity of Ariane 5 was too high to be represented as a 16-bit integer
- Error handling was suppressed for performance reasons

\*Source: <http://moscova.inria.fr/~levy/talks/10enslongo/enslongo.pdf>

```
-- Vertical velocity bias as measured by sensor  
L_M_BV_32 :=  
    TDB.T_ENTIER_32S ((1.0/C_M LSB_BV) *  
        G_M_INFO_DERIVE(T_ALG.E_BV));  
-- Check, if measured vertical velocity bias can be  
-- converted to a 16 bit int. If so, then convert  
if L_M_BV_32 > 32767 then  
    P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;  
elseif L_M_BV_32 < -32768 then  
    P_M_DERIVE(T_ALG.E_BV) := 16#8000#;  
else  
    P_M_DERIVE(T_ALG.E_BV) :=  
        UC_16S_EN_16NS(TDB.T_ENTIER_16S(L_M_BV_32));  
end if;  
-- Horizontal velocity bias as measured by sensor  
-- is converted to a 16 bit int without checking  
P_M_DERIVE(T_ALG.E_BH) :=  
    UC_16S_EN_16NS (TDB.T_ENTIER_16S ((1.0/C_M LSB_BH) *  
        G_M_INFO_DERIVE(T_ALG.E_BH)));
```

Testy nigdy nie dadzą pełnej gwarancji



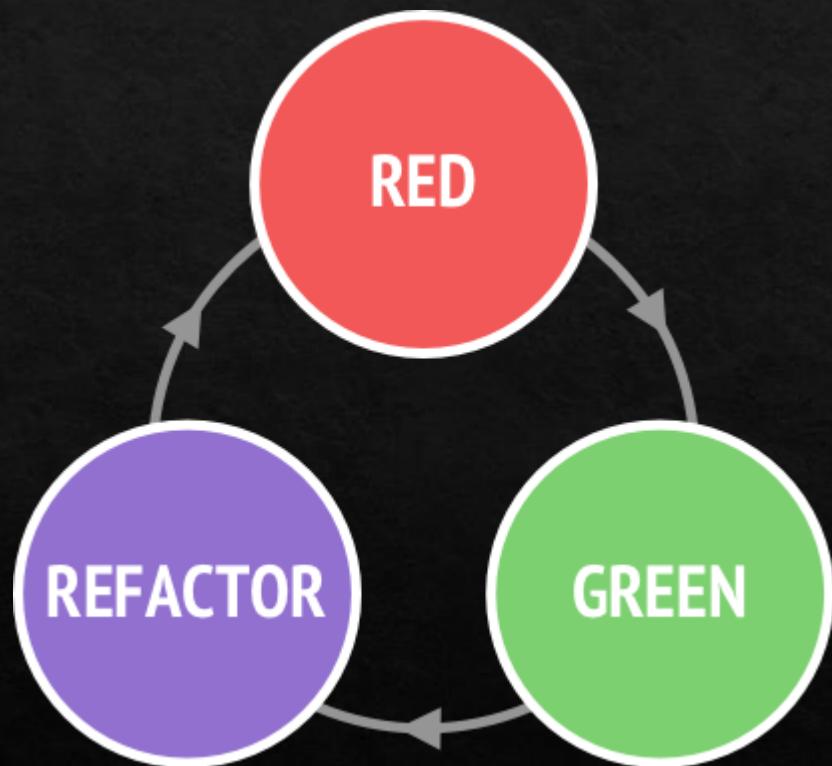
Testy stanowią ważną część architektury całego systemu

# TDD – Test Driven Development

(Rozwój sterowany testami)

Najpierw piszemy testy, a dopiero potem kod

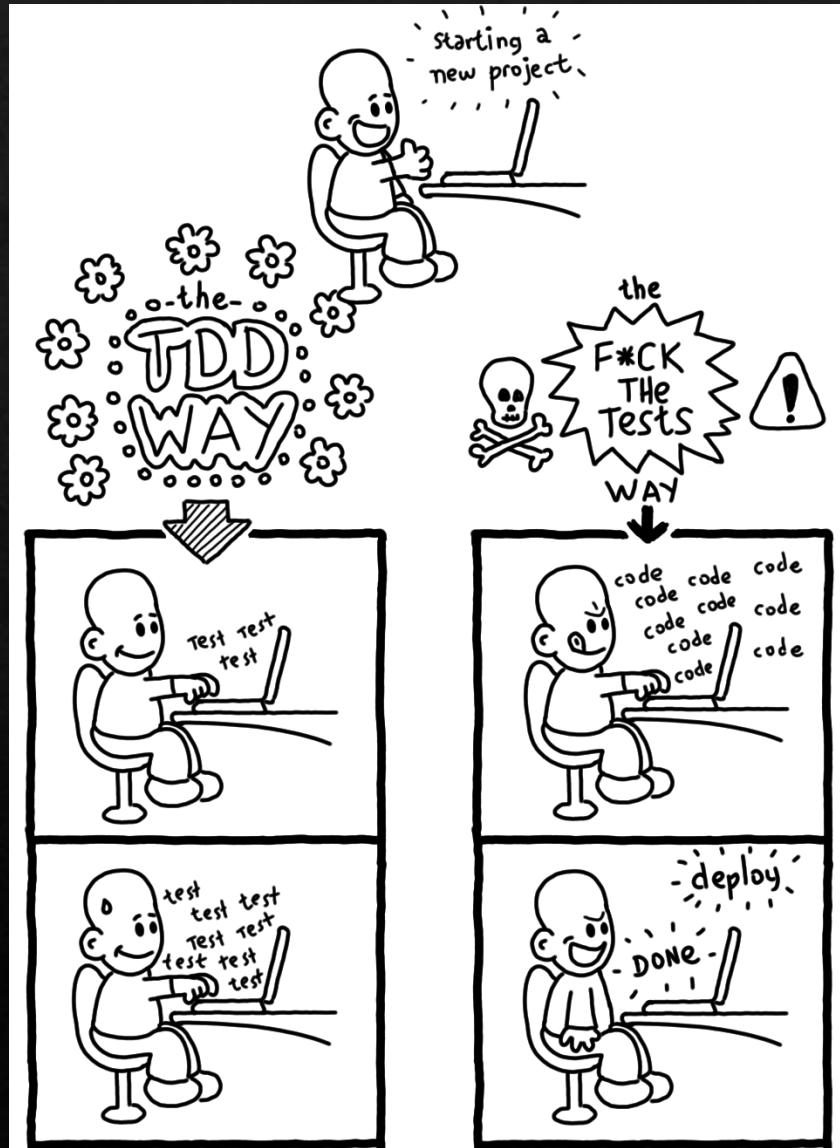
# TDD – Test Driven Development



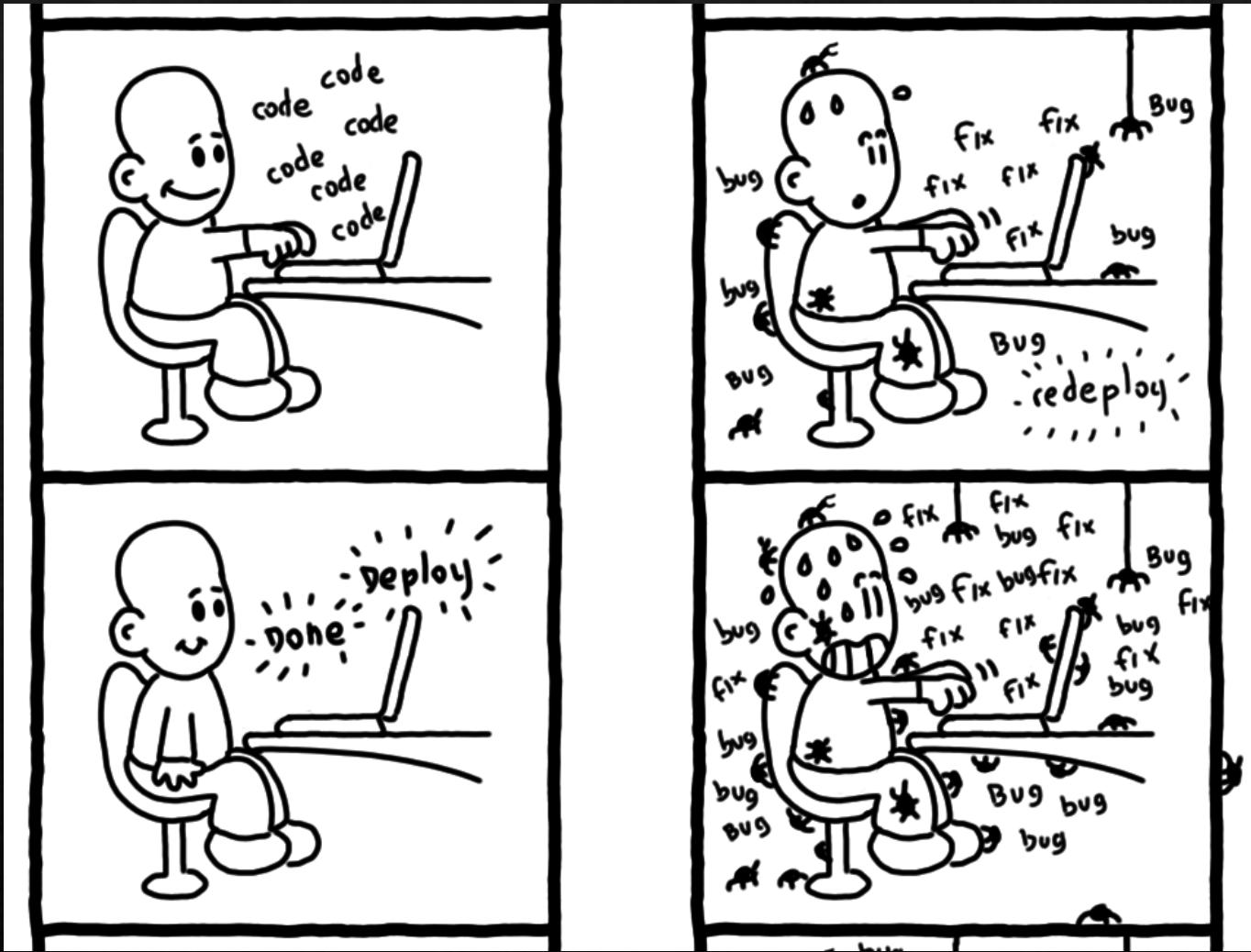
# Metodyka TDD

- ❖ Nie można zacząć pisać kodu wykonywalnego do momentu napisania niespełnionego testu jednostkowego.
- ❖ W danej chwili nie należy mieć więcej niż jeden działający test jednostkowych.
- ❖ Nie można pisać większej ilości kodu wykonywalnego, niż jest to potrzebne do spełnienia obecnie niespełnionego testu.

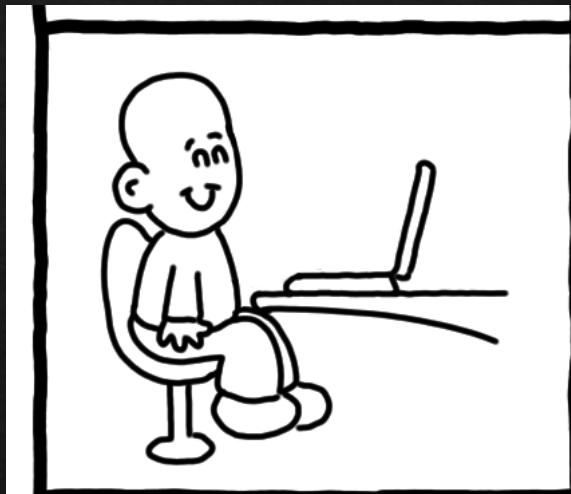
# TDD – Test Driven Development



# TDD – Test Driven Development



# TDD – Test Driven Development



aniel Stori {turnoff.us}



**Strukturalne**

**Jednostkowe**

**Manualne**

**Eksploracyjne**

**Regresyjne**

**Akceptacyjne**

**Białej skrzynki**

**Mefunkcjonalne**

**Dynamiczne**

**Stacyczne**

**Ręczne**

**Poczytalności**

**Systemowe**

**Wydajnościowe**

**Dymne**

**Bezpieczeństwa**

**Funkcjonalne**

**Dostępności**

**Podatności**

**Integracyjne**

**Automatyczne**

**Czarnej skrzynki**

# Podstawowe rodzaje testów

- ❖ Testy manualne

# Podstawowe rodzaje testów

- ❖ Testy manualne
- ❖ Testy automatyczne

# Rodzaje testów

- ❖ Testy ręczne aka manualne (manual tests)

# Rodzaje testów

- ❖ Testy ręczne aka manualne (manual tests)
  - ❖ Testy eksploracyjne (exploratory tests)

# Rodzaje testów

- ❖ Testy ręczne aka manualne (manual tests)
  - ❖ Testy eksploracyjne (exploratory tests)
- ❖ Testy automatyczne (automated tests)

# Rodzaje testów

- ❖ Testy ręczne aka manualne (manual tests)
  - ❖ Testy eksploracyjne (exploratory tests)
- ❖ Testy automatyczne (automated tests)
  - ❖ Testy jednostkowe (unit tests)

# Testy jednostkowe



# Rodzaje testów

- ❖ Testy ręczne aka manualne (manual tests)
  - ❖ Testy eksploracyjne (exploratory tests)
- ❖ Testy automatyczne (automated tests)
  - ❖ Testy jednostkowe (unit tests)
- ❖ Testy integracyjne (układ hamulcowy)

# Rodzaje testów

- ❖ Testy ręczne aka manualne (manual tests)
  - ❖ Testy eksploracyjne (exploratory tests)
- ❖ Testy automatyczne (automated tests)
  - ❖ Testy jednostkowe (unit tests)
  - ❖ Testy integracyjne (układ hamulcowy)
  - ❖ Testy akceptacyjne (dział kontroli jakości)

# Rodzaje testów

- ❖ Testy ręczne aka manualne (manual tests)
  - ❖ Testy eksploracyjne (exploratory tests)
- ❖ Testy automatyczne (automated tests)
  - ❖ Testy jednostkowe (unit tests)
  - ❖ Testy integracyjne (układ hamulcowy)
  - ❖ Testy akceptacyjne (dział kontroli jakości)
  - ❖ Testy systemowe (E2E) (użytkownik końcowy, kierowca)

# F.I.R.S.T – cechy dobrze napisanych testów jednostkowych

- ❖ **F**ast (szybkie)
- ❖ **I**ndependent (niezależne)
- ❖ **R**epeatable (powtarzalne)
- ❖ **S**elf-validating (samokontrolujące)
- ❖ **T**imely (na czas)

Dobra wiadomość: każdy z nas stworzył już co najmniej kilka testów, nawet jeżeli o tym nie wie