SLAJD1:

Celem testowania jest wykrycie błędów oraz ocena niezawodności oprogramowania

Rozróżnia się następujące terminy: Weryfikacja (verification) - testowanie zgodności systemu z wymaganiami zdefiniowanymi w fazie określenia wymagań. Atestowanie (validation) - ocena systemu lub komponentu podczas lub na końcu procesu jego rozwoju na zgodności z wyspecyfikowanymi wymaganiami.

SLAJD2:

**Testy statyczne** – forma testowania oprogramowania bez uruchamiania programu podczas testów. Test polega na automatycznym i ręcznym sprawdzaniu kodu w celu znalezienia błędów. Najczęściej wykonywany jest przez twórców kodu jako pierwsze i podstawowe sprawdzenie każdego programu. Testowanie statyczne sprawdza podstawową poprawność kodu i pozwala ocenić, czy program jest gotowy na bardziej szczegółowe testowanie.

W testowaniu statycznym można wyróżnić podstawową analizę kodu i precyzyjne wyszukiwanie typowych błędów.

WYSZUKIWANIE BLEDOW: - to nie iwemczyktosbedzie mialczyco, jaknie to strescic to cojestnapisane nizej

Wyszukiwanie typowych błędów jest najczęściej dokonywane przez programistów poprzez dokładne czytanie ze zrozumieniem całego kodu programu w celu wykrycia typowych błędów, które nie są wykrywane przez narzędzia programistyczne, natomiast mogą doprowadzić do błędnych wykonań w programie. Przykładowymi takimi błędami są odwołania do pozycji o wartości przekraczającej zakres tablicy. Wyszukiwanie typowych błędów, często mało doceniane jako metoda testowania, potrafi mieć znaczący wpływ na skuteczność procesu testowania poprzez wczesne wykrycie skomplikowanych błędów u źródła.

SLAJD 3

ANALIZA KODU

**Analiza statyczna kodu** - analiza struktury kodu źródłowego lub kodu skompilowanego bez jego uruchomienia.

Static Analysis Software Testing (SAST) - narzędzia do automatycznej analizy statycznej

Analiza statyczna a inspekcja kodu („code review”)

SLAJD 5:

PEP 8

dokument zawierający wytyczne i najlepsze praktyki dotyczące sposobu pisania kodu Pythona. Został napisany w 2001 r. Przez Guido van Rossum, Barry Warsaw i Nicka Coghlana. Głównym celem PEP 8 jest poprawa czytelności i spójności kodu Pythona.

PEP oznacza Python Enhancement Proposal, a jest ich kilka. PEP to dokument opisujący nowe funkcje proponowane w Pythonie i dokumentujący aspekty Pythona, takie jak projektowanie i styl, dla społeczności.

może być trudno zapamiętać, co kawałek kodu robi kilka dni lub tygodni po jego napisaniu. Jeśli śledzisz PEP 8, możesz być pewien, że dobrze określiłeś swoje zmienne.

Skomentowałeś swojego kodu oraz uzycie whitespace sprawia, że ​​ kod jest bardziej czytelny i łatwiejszy do powrotu.

Przestrzeganie zasad PEP 8 może uczynić naukę Pythona znacznie przyjemniejszym zadaniem.

Obserwowanie PEP 8 jest szczególnie ważne w pracy programisty. Pisanie czytelnego kodu pokazuje profesjonalizm oraz oszczędza wiele czasu. Powie pracodawcy, że rozumiesz, jak dobrze zorganizować swój kod.

SLAJD 6 NAMING STYLES:

Oto niektóre z typowych konwencji nazewnictwa i przykłady ich użycia. Aby jednak napisać czytelny kod, nadal musisz uważać na wybór liter i słów. Oprócz wyboru poprawnych stylów nazewnictwa w kodzie, musisz także ostrożnie wybrać nazwy. TUTEJ znajduje się kilka wskazówek, jak to zrobić tak skutecznie, jak to możliwe.