
# Занятие 28.
# Тестирование

1

# Основы тестирования

# Тестирование

**Тестирование** − это процесс проверки программного обеспечения с целью обнаружения ошибок, дефектов или неправильного поведения системы. Оно выполняется для уверенности в качестве программы и для обеспечения ее правильной работы.

# Тестирование

1. Обнаружение ошибок
2. Обеспечение качества - соответсвие требованиям
3. Проверка функциональности- корректность работы
4. Предотвращение проблем - тестирование потенциальных проблем
5. Улучшение надежности - стабильность
6. Экономия времени и ресурсов - находим проблемы на ранних этапах разработки

2

# Виды тестирования

# Виды тестирования

- **Модульное тестирование**: Это тестирование отдельных модулей или компонентов программного обеспечения для проверки их функциональности в изоляции от других частей системы (напр, ф-ции).
- **Интеграционное тестирование**: В данном виде тестирования проверяется взаимодействие между различными модулями или компонентами программного обеспечения, чтобы убедиться, что они корректно взаимодействуют и передают данные друг другу.
- **Системное тестирование**: Этот вид тестирования выполняется на уровне всей системы для проверки ее соответствия требованиям, функциональности и производительности. Обычно включает проверку различных сценариев использования и возможностей системы

# Виды тестирования

- **End-to-end**: Это тестирование, выполняемое с целью подтверждения, что система или продукт готов к внедрению и использованию конечными пользователями. Обычно проводится на основе предварительно определенных критериев.
- **Функциональное тестирование**: Заключается в проверке функциональности программного обеспечения, чтобы убедиться, что оно выполняет заданные функции и соответствует требованиям.
- **Нагрузочное тестирование**: Цель этого тестирования - проверить производительность и стабильность системы при нагрузке, симулирующей реальные рабочие условия.
- **Стресс-тестирование**: В данном виде тестирования проверяется поведение системы при превышении нормальной нагрузки или в экстремальных условиях, чтобы определить ее пределы и выявить проблемы с производительностью или стабильностью.

# Виды тестирования

- **Тестирование безопасности:** Этот вид тестирования выполняется для выявления уязвимостей и проверки защищенности системы от возможных атак или несанкционированного доступа.
- **Тестирование совместимости**: Заключается в проверке работоспособности программного обеспечения на различных платформах, операционных системах, браузерах или устройствах.
- **Тестирование пользовательского интерфейса**: В этом виде тестирования проверяется удобство использования и соответствие пользовательского интерфейса ожиданиям и требованиям пользователей.
- **Регрессионное тестирование** выполняется для проверки уже протестированных функций или модулей программного обеспечения после внесения изменений или добавления нового кода. Цель - убедиться, что внесенные изменения не привели к появлению новых ошибок или не повлияли на уже работающую функциональность.

# Unit тесты

**Unit-тесты** (также известные как модульные тесты) - это вид тестирования программного обеспечения, в котором отдельные компоненты или модули (например, функции, классы, методы) тестируются на корректность и правильность работы в изоляции от остальной системы.

**Цель unit-тестирования** состоит в проверке правильности работы отдельных компонентов программного обеспечения, их взаимодействия и соответствия ожидаемому поведению. Unit-тесты помогают выявить ошибки и дефекты на ранних стадиях разработки, что улучшает качество и надежность программного обеспечения. Они также облегчают рефакторинг кода, поскольку позволяют быстро проверить, что изменения не нарушили работу существующего функционала.

# Unit тесты

1. **Изоляция**: Unit-тесты выполняются в изоляции от других компонентов или модулей системы. Они проверяют функциональность конкретного компонента независимо от других частей системы.
2. **Малый объем**: Unit-тесты фокусируются на небольших единицах кода, таких как отдельные функции или методы. Это позволяет более точно определить, какие части программы требуют тестирования и где возможно обнаружение ошибок.
3. **Автоматизация**: Unit-тесты обычно автоматизируются с помощью тестовых фреймворков и инструментов, что позволяет легко запускать и повторять тесты в разных сценариях без необходимости ручного вмешательства.
4. **Быстрота выполнения:** Unit-тесты выполняются быстро, поскольку тестируют отдельные компоненты небольшого объема кода. Это позволяет быстро обнаруживать ошибки и проводить регулярное тестирование в процессе разработки.

2

# unittest vs pytest

# pytest vs unittest

**unittest и pytest -** это два популярных тестовых фреймворка для написания и выполнения тестов в Python. Они имеют различные подходы к организации и запуску тестов, а также предлагают различные функциональные возможности

# Синтаксис и струтура кода

- **unittest:** В unittest используется классический стиль объектно-ориентированного программирования, где тестовые методы определяются внутри классов, наследующихся от базового класса unittest.TestCase. Тестовые методы должны начинаться с префикса "test_".
- **pytest:** pytest использует простой и понятный синтаксис, который не требует явного наследования от базового класса. Тестовые функции могут быть написаны без использования классов и могут иметь любое имя (не обязательно должны начинаться с "test_").

# Удобство и гибкость

- **unittest:** unittest предоставляет широкий набор функций и возможностей, таких как использование ассертов, параметризованные тесты, настраиваемые итераторы для тестовых данных и многое другое. Он также включает в себя средства для организации и запуска тестовых наборов.
- **pytest:** pytest предоставляет простой и гибкий интерфейс для написания тестов, поддерживая широкий набор плагинов и расширений для различных тестовых сценариев. Он обладает богатыми возможностями, такими как автоматическое обнаружение и запуск тестов, параметризация тестовых функций, маркировка тестов для выборочного выполнения и т. д.

# Интеграция и сообщество

- **unittest**: unittest является стандартной библиотекой Python, что означает, что он встроен в сам Python и не требует установки дополнительных пакетов. Он имеет документацию и обширное сообщество пользователей.

- **pytest**: pytest является сторонней библиотекой, которую необходимо установить отдельно с помощью pip. Он также имеет обширную документацию и активное сообщество пользователей, а также поддерживает интеграцию с другими инструментами и плагинами.

# Сложность

- **unittest:** unittest обладает большим количеством функций и возможностей, что может сделать его более сложным для начинающих. Некоторые пользователи считают, что он требует больше кода для написания тестов.

- **pytest:** pytest предоставляет более простой и интуитивный подход к написанию тестов, что может быть удобнее для начинающих. Он также предлагает множество сокращений и упрощений синтаксиса, которые позволяют писать более читаемые и компактные тесты.

# Фичи pytest

- **Параметризация тестовых функций**: Pytest позволяет параметризовать тестовые функции, что позволяет выполнить один и тот же тест с различными входными данными или аргументами.
- **Фикстуры**: Pytest предлагает концепцию фикстур, которые позволяют определить и переиспользовать ресурсы, настройки и предварительные условия для тестов. Фикстуры могут быть определены на уровне функций, модулей, классов или даже всего проекта.
- **Маркировка тестов:** Pytest позволяет маркировать тесты с помощью пользовательских меток (маркеров), что позволяет выборочно запускать или исключать определенные группы тестов. Это полезно, например, для разделения тестов по категориям, уровню сложности или типу окружения.

# Фичи pytest

- **Параметризация и конфигурация:** Pytest поддерживает различные варианты параметризации и настройки выполнения тестов, такие как использование файлов конфигурации, командной строки, переменных окружения и других способов.
- **Обширные возможности по ассертам:** Pytest предлагает широкий спектр встроенных ассертов, которые позволяют проверять различные условия и сравнивать значения. Он также поддерживает кастомные ассерты для более сложных проверок.
- **Поддержка множества плагинов:** Pytest имеет богатую экосистему плагинов, которые позволяют расширить его функциональность и интегрировать его с другими инструментами и фреймворками.
- **Поддержка параллельного выполнения:** Pytest имеет возможность параллельного выполнения тестов, что позволяет ускорить процесс тестирования в многоядерных системах.

# Спасибо за внимание!