Automation tester

Курс “Автоматизация тестирования”

Зачем автоматизировать тестирование?

Зачем автоматизировать тестирование?

- Время тестирования
- Человеческий фактор
- Отчетность
- CI/CD*

PS: CI/CD
(Continuous Integration, Continuous Delivery)
“Непрерывная интеграция”, “Непрерывная доставка”

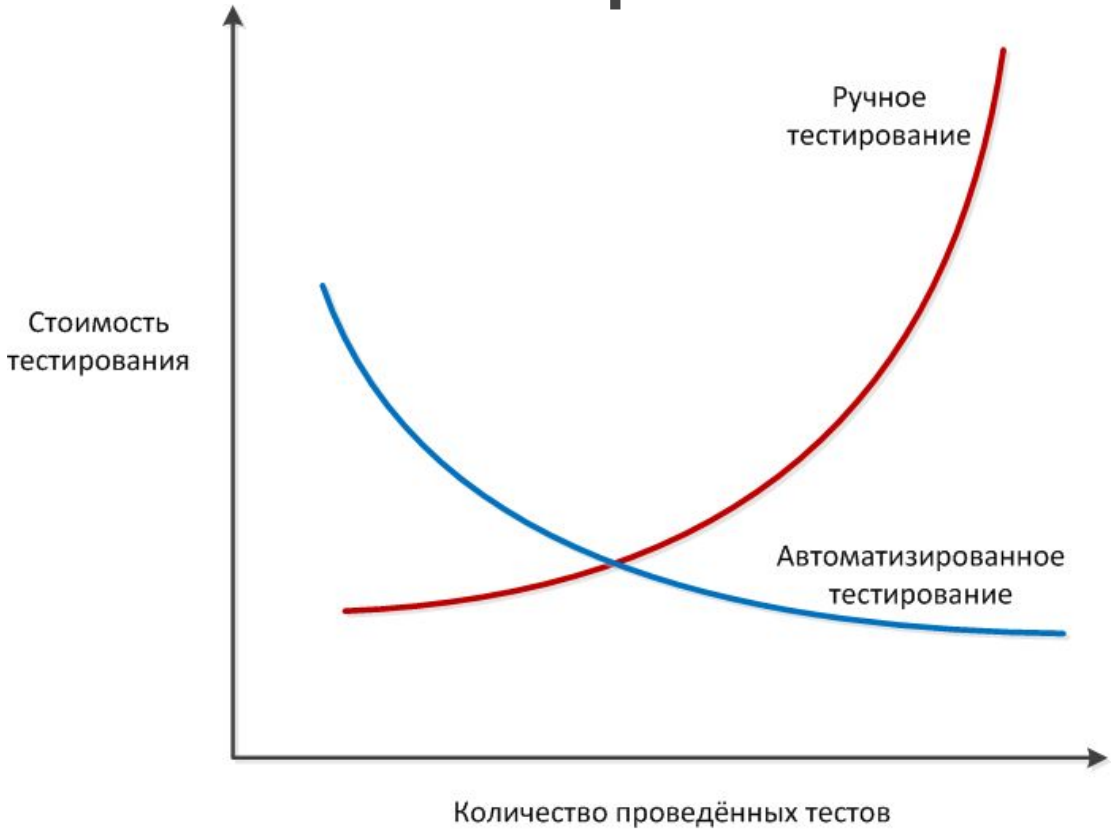
Плюсы автоматизации

- **Повторяемость.** Код автотестов может быть использован неоднократно, особенно при внедрении новой функциональности.
- **Временной фактор.** Ручное тестирование – это долгий и ресурсоемкий процесс, в то время как код для сценария пишется один раз.
- **Нагрузка на приложение.** Когда используется автоматизированное тестирование, становится возможным моделирование большой нагрузки, которая приближена к реальной ситуации.
- **Высокое покрытие тестами**
- **Быстрое обнаружение ошибок**

Минусы автоматизации

- **Стоимость.** Для небольших проектов инструменты автоматизированного тестирования могут оказаться достаточно затратными, поэтому более рационально их использовать для долгосрочных проектов.
- **Отсутствие тестирования глазами пользователя.** Автотест гарантирует отсутствие дефектов только в том месте где он проверяет, абсолютно без погрешностей. Человек может идти по одному сценарию и найти много дефектов не связанных со сценарием, робот о них промолчит.
- **переоценка автотестов.** Часто качество автотестов переоценивают. Пройденные тесты легко воспринимают как доказательство отсутствия дефектов, это не так.

Стоимость тестирования



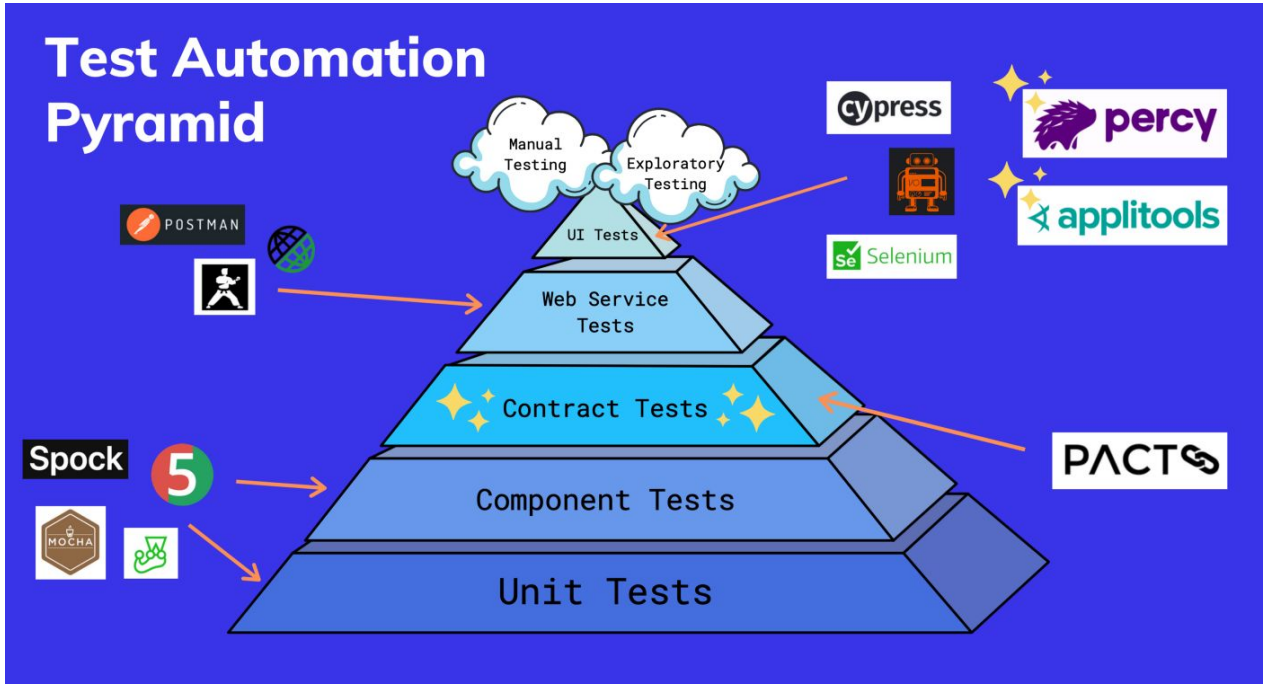
что можно автоматизировать

- Модульное тестирование
- Регрессионные и интеграционные тесты
- Нагрузочные тесты и тесты производительности
- Базовая функциональность (smoke)

Какие тесты НЕ нужно автоматизировать

- Пользовательский опыт (UX)
- Стадии ранней разработки
- Функциональность, не имеющая большой важности
- Тесты без понятных результатов
- Тесты, которые невозможно полностью автоматизировать

Фреймворки и выбор фреймворка тестирования



выбор фреймворка тестирования

- Простота разработки и исполнения скриптов (поддержка гибких процессов и коротких итераций)
- Соответствие приложения языку разработки теста
- Отсутствие пробелов в функциональных возможностях тестирования
- отчетность
- варианты запуска

Практика

Python

Python — это язык общего назначения. Используется в науке, разработке ПО, автоматизации и т.д.

ООП

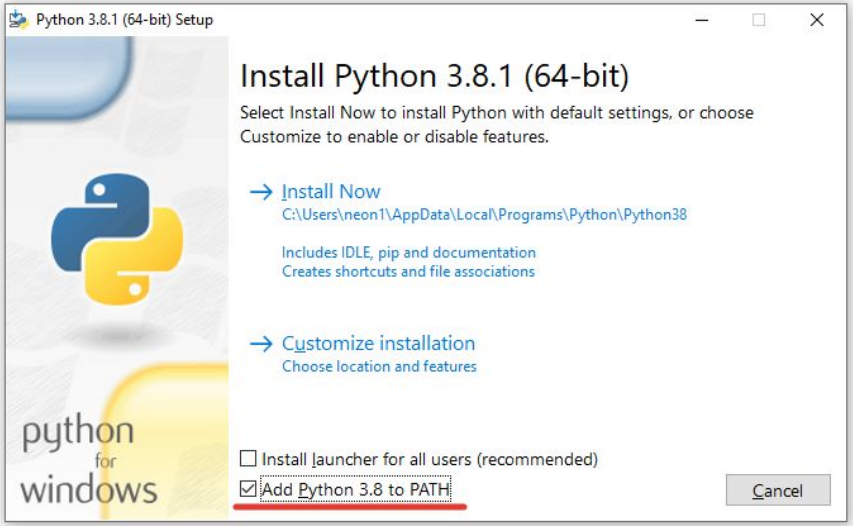
С одной стороны, Python – объектно-ориентированный язык. Вся мощь объектно-ориентированного подхода доступна программисту Python, но с другой стороны – Python не вынуждает программиста всегда использовать ООП.

интерпретируемый

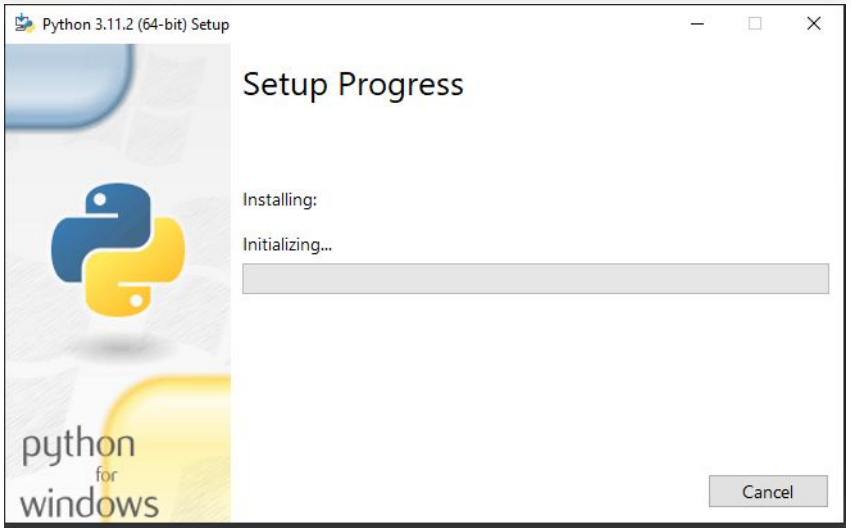
Что это означает – интерпретируемый? А то, что код выполняется (интерпретируется) из исходного текста, без предварительного перевода в машинный код. Код, написанный на компилируемых языках, типа C++, сначала переводятся в машинный код (самый распространенный пример – откомпилированного кода – файл с расширением .exe).

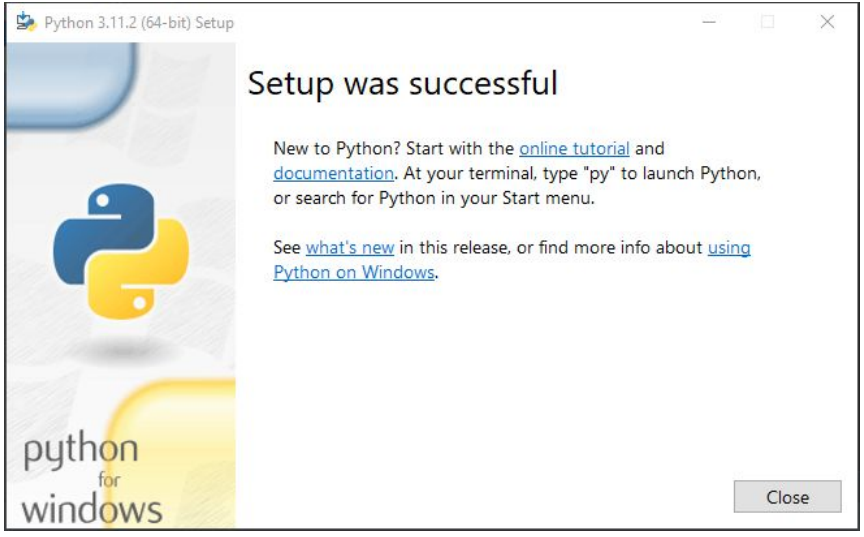
Установка Python3 PIP3

- перейти на страницу https://www.python.org/downloads/
- Загрузите последнюю версию Python
- Если сайт предлагает ОС отличную от Вашей выберите ее из предложенных, далее загрузите Python
- После загрузки .exe файла откройте его.



- Ставим галочку возле Add Python 3.x to PATH
- Нажимаем Install Now
- Процесс установки запущен
- После установки нажимаем Close

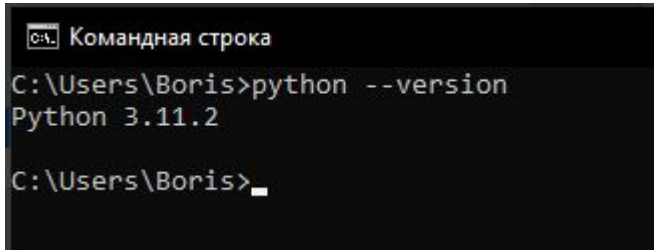




Проверка

Для проверки установки Python:

- откройте пуск
- введите в поиск cmd
- выберите “командная строка”
- в появившемся окне введите команду `python --version`
- Вывод будет примерно следующий:



пакетный менеджер

пакетный менеджер (Система управления пакетами) - набор программного обеспечения, позволяющего управлять процессом установки, удаления, настройки и обновления различных компонентов программного обеспечения. Системы управления пакетами активно используются в различных дистрибутивах операционной системы Linux и других UNIX-подобных операционных системах.

Репозиторий — это место, где хранятся пакеты. Проще говоря, какие-то чужие компьютеры.

Обычно у каждого семейства операционных систем или ЯП свой репозиторий и свой менеджер пакетов, который с ним работает.

Мы будем использовать пакетный менеджер **PIP**

pip

pip - стандартный менеджер пакетов для Python. Он позволяет устанавливать и управлять дополнительными пакетами, которые не являются частью стандартной библиотеки Python.

PyCharm

PyCharm — это среда программирования для языка Python, или IDE. Средами называют программы, в которых можно писать, запускать и отлаживать код, устанавливать новые расширения и дополнительные модули. Это очень мощный инструмент.

Установка PyCharm

https://www.jetbrains.com/ru-ru/pycharm/download/#section=windows

https://www.jetbrains.com/ru-ru/pycharm/download/#section=mac

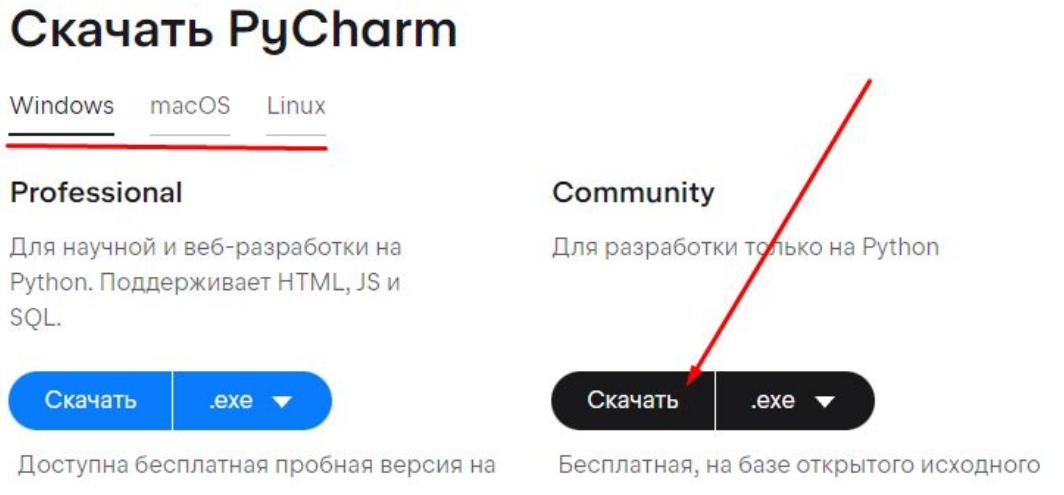
https://www.jetbrains.com/ru-ru/pycharm/download/#section=linux

Скачиваем установщик **PyCharm Community** для своей ОС с официального сайта JetBrains.

В самой установке ничего особенного нету. По сути только нажимаем на кнопки next, и в завершение на кнопку Install.

Версия: 2022.3.2
Сборка: 223.8617.48
26 января 2023 г.

Системные требования

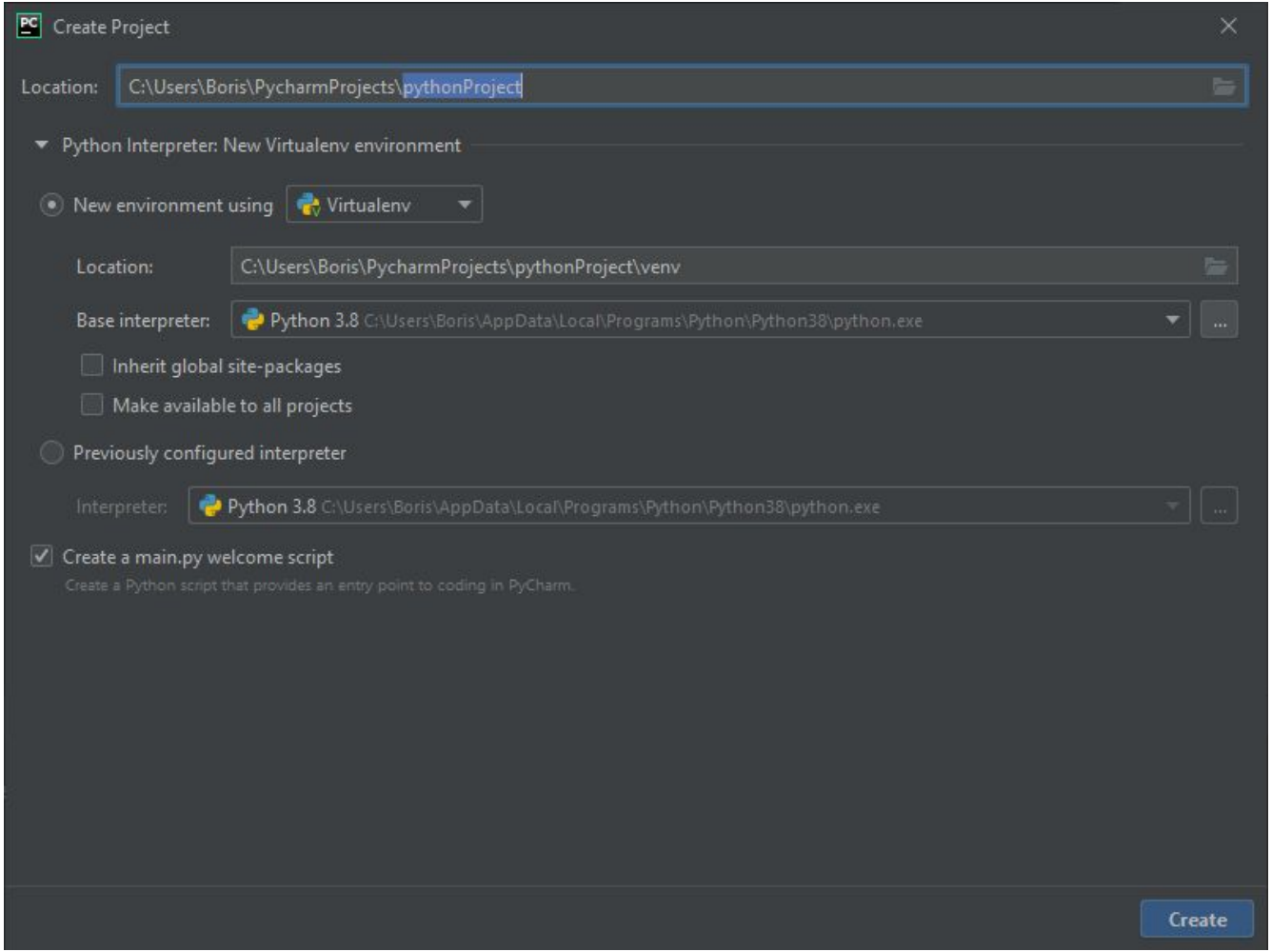


Создание проекта

Запускаем PyCharm и окне приветствия выбираем **Create New Project.**

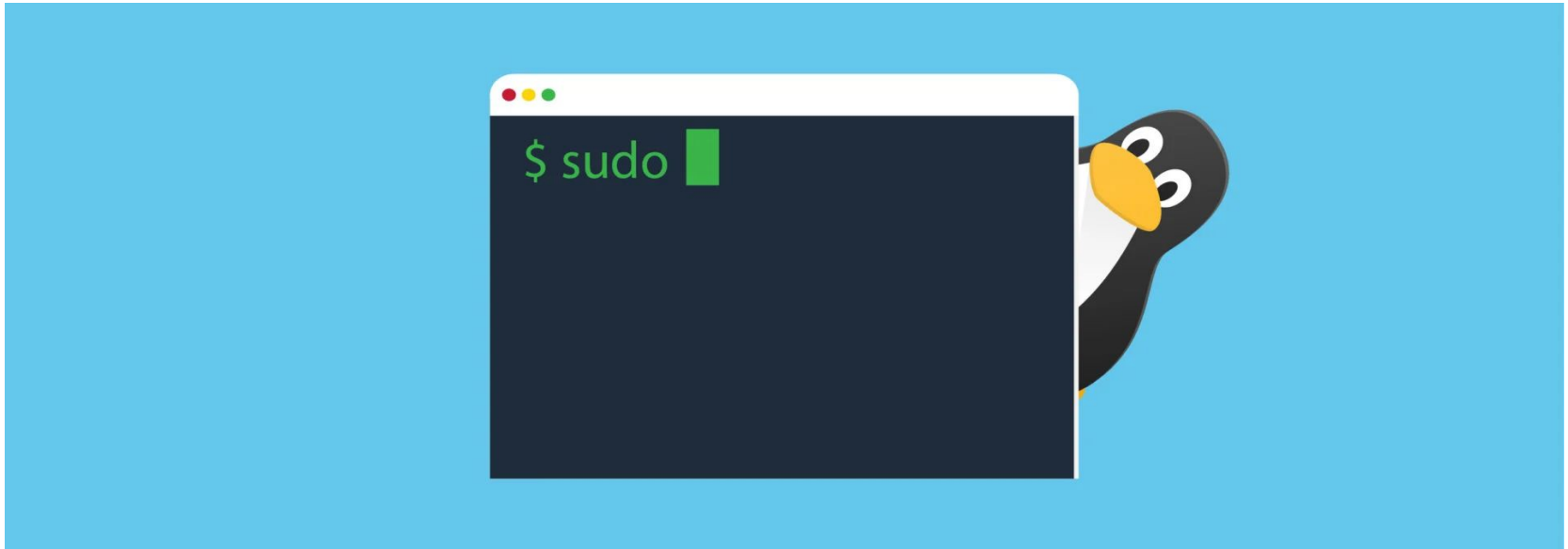
В мастере создания проекта, указываем в поле **Location** путь расположения создаваемого проекта. Имя конечной директории также является именем проекта указываем **ITMO_Automation**. В примере директория называется **'pythonProject'**.

Нажимаем на Create.


Create Project
Location:
C:\Users\Boris\PycharmProjects\pythonProject
Python Interpreter: New Virtualenv environment
New environment using
Virtualenv
Location:
C:\Users\Boris\PycharmProjects\pythonProject\venv
Base interpreter:
Python 3.8 C:\Users\Boris\AppData\Local\Programs\Python\Python38\python.exe
Inherit global site-packages
Make available to all projects
Previously configured interpreter
Interpreter:
Python 3.8 C:\Users\Boris\AppData\Local\Programs\Python\Python38\python.exe
Create a main.py welcome script
Create a Python script that provides an entry point to coding in PyCharm.
Create

Установка зависимостей - через терминал

Терминал – это окно коммуникации между человеком и операционной системой.



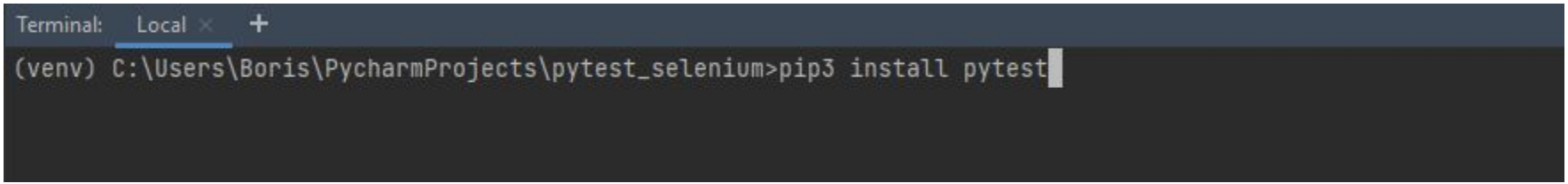
Терминал

Терминал – это

Начнем с понимания значения слова «терминал». Терминал – это часть некой системы, обеспечивающая ее взаимодействие с внешней средой. Например, терминал является частью аэропорта, где производятся операции с пассажирами перед их отправкой в самолет или после прибытия самолета в аэропорт. То есть, терминал аэропорта – это точка входа и выхода в системе авиаперевозок, в которой персонал задает параметры для операций с пассажирами: кто, когда и куда отправляется. То же самое делает терминал в компьютере, только вместо пассажиров – данные, а вместо персонала – пользователь. Таким образом, терминал – это окно коммуникации между человеком и операционной системой.

Установка PyTest

- > pip3 install pytest



Создания файла зависимостей

у Python есть возможность хранить все установленные зависимости в одном месте. Тем самым мы можем публиковать в GIT только наш код, а все либы докачивать при работе с ним.

`pip3 freeze > requirements.txt`	запись зависимостей
`pip3 freeze`	вывод в консоль всех зависимостей
`pip3 install -r requirements.txt`	установка всех зависимостей

Требования к названиям

1. Имя файла должно начинаться с ”**test**” или заканчиваться “**test.py**”.
2. Имена функций и переменных должны быть написаны в нижнем регистре, а слова должны быть разделены подчеркиванием. При этом имя тестовой функции должно начинаться с “**test_**”, например “**test_check_input**”.

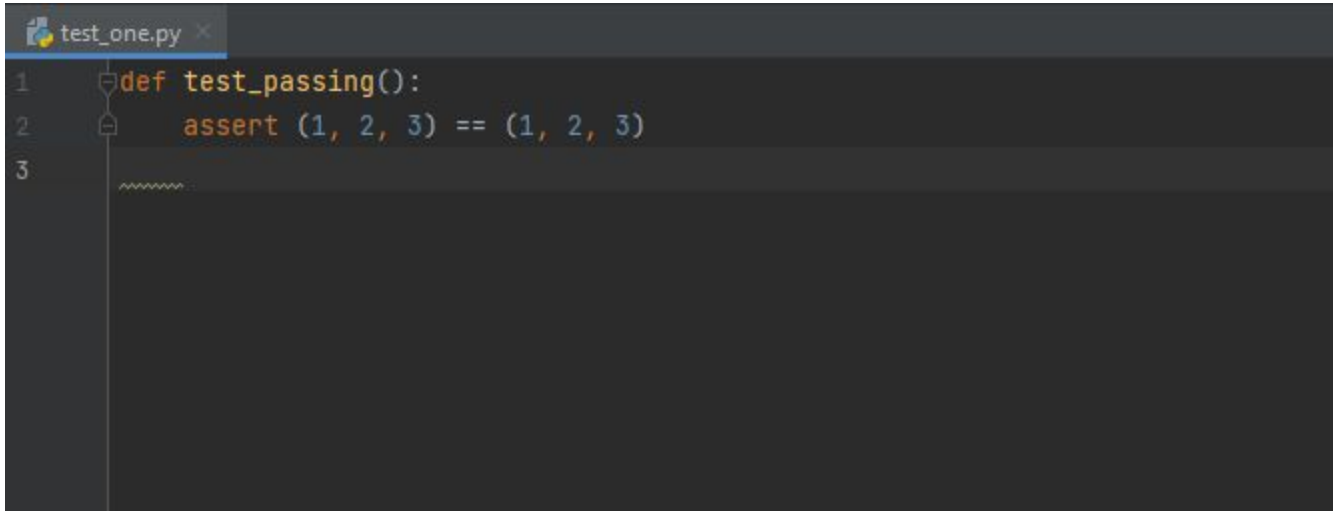
PS: **Snake case** относится к стилю письма, в котором каждый пробел заменяется символом подчеркивания (_), а первая буква каждого слова пишется строчными буквами. Это обычно используемое соглашение об именах в вычислительной технике , например, для имен переменных и подпрограмм , а также для имен файлов .

Первый тест

создайте директорию “**tests**”

в ней создайте файл “**test_one.py**”

запишите функцию



Запуск первого теста

- откройте терминал
- введите “***pytest***”

Без аргументов pytest исследует ваш текущий каталог и все подкаталоги для тестовых файлов и запустит тестовый код, который найдёт. Если вы передадите pytest имя файла, имя каталога или список из них, то будут найдены там вместо текущего каталога. Каждый каталог, указанный в командной строке, рекурсивно исследуется для поиска тестового кода.

```
(venv) C:\Users\Boris\PycharmProjects\PyTest-1\tests_task_1>pytest test_one.py
============================================================================================================ test session starts =============
platform win32 -- Python 3.8.8, pytest-7.2.0, pluggy-1.0.0
rootdir: C:\Users\Boris\PycharmProjects\PyTest-1\tests_task_1
collected 1 item

test_one.py .

============================================================================================================= 1 passed in 0.01s ==============

(venv) C:\Users\Boris\PycharmProjects\PyTest-1\tests_task_1>
```

команды pytest

pytest –help	показывает список команд
pytest	запускает все тесты в текущем каталоге
pytest “каталог/файл”	запускает все тесты в каталоге/файле
pytest -v	дает расширенный отчет

команды терминала

cd “каталог”	Перейти в папку
cd ../	Перейти в папку выше
dir	Выводит список файлов и папок в текущей папке
mkdir “название новой папки”	Создать папку
tree	Показать дерево папок

Зависимости

`pip3 freeze > requirements.txt`	запись зависимостей
`pip3 freeze`	вывод в консоль всех зависимостей
`pip3 install -r requirements.txt`	установка всех зависимостей

Спасибо за внимание