Самое важное

Инструкции по установке Python и PyCharm

Работа с командной строкой
Командная строка является альтернативным способом запуска программ, которые пишут на Python.

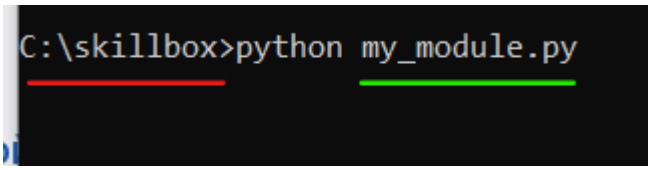
Для запуска программы: указываем, через что мы хотим запустить файл с кодом, а затем указываем путь к самому файлу с кодом.

python my_module.py

Важно понимать две вещи.

1. Python — это ссылка на python.exe в директории, в которую вы установили Python. Обычно эта директория автоматически добавляется в переменную среды PATH, поэтому можно использовать команду python из любой открытой директории в командной строке. Однако если директория с Python не добавилась в эту переменную, нужно сделать это вручную (или переустановить Python, поставив галочку рядом с add to PATH). Можно использовать этот или подобные гайды по добавлению python в PATH.

2. Путь до модуля надо указывать либо полностью от корня (C:\skillbox\my_module.py), либо относительно текущей директории командной строки. Например, если командная строка открыта в папке skillbox, можно просто указать название модуля:

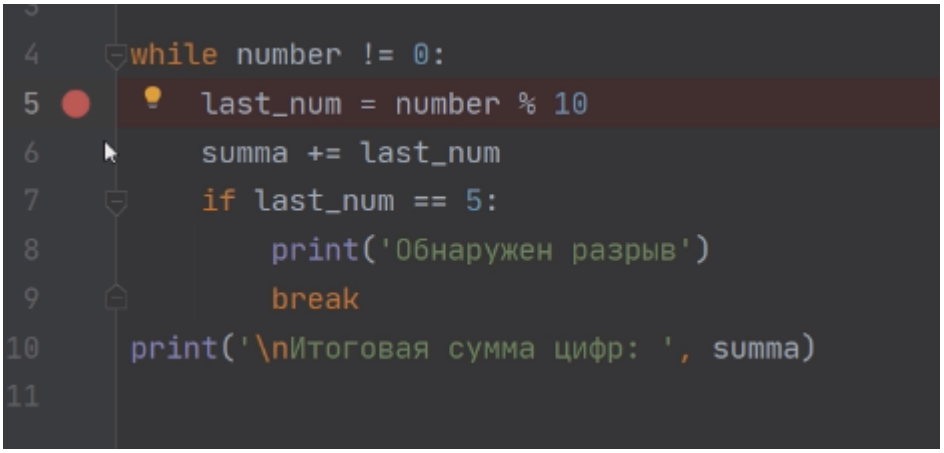


Базовые команды для работы с командной строкой (терминалом):
https://drive.google.com/file/d/1mowpka1hjeA9eqJV2vPS0MJqx_Y56zzh/view?usp=sharing

Отладка программ в PyCharm
PyCharm (и другие IDE - Интегрированная среда разработки (англ. Integrated Development Environment)) позволяют использовать дополнительные инструменты по работе с кодом. Один из таких инструментов — дебаггер (debugger). Он позволяет выполнять код построчно и в процессе выводит дополнительную информацию о текущем состоянии программы (например, можно посмотреть значения всех переменных в текущий момент выполнения кода).

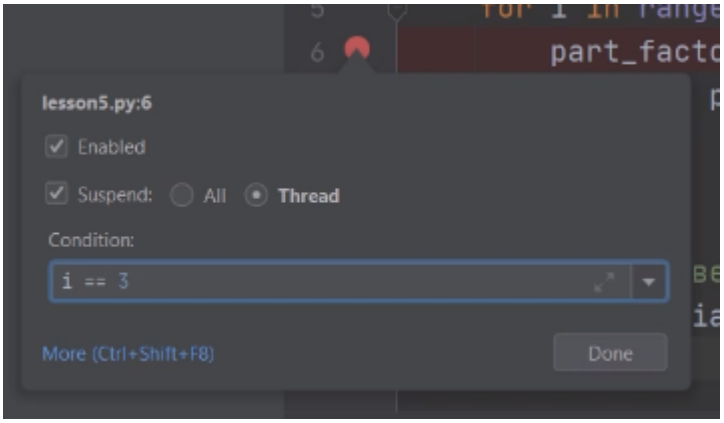
Для запуска нужно выполнить два действия.

1. Указать точку остановы — строку, на которой выполнение программы будет поставлено на паузу и после которой можно выполнять код построчно. Как это выглядит:



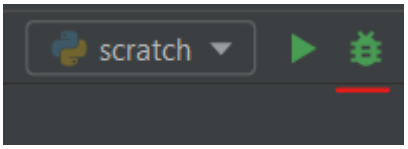
Выбираете строку, с которой хотите проверить работу программы, и щёлкаете мышкой по полю слева от этой строчки. Строка будет выделена красным цветом, а на поле появится красная точка.

При необходимости можно установить множество таких точек. Можно даже изменить принцип работы этой точки, добавив условие, при котором остановка выполнения сработает. Сделать это можно, нажав на красную точку правой кнопкой мыши:

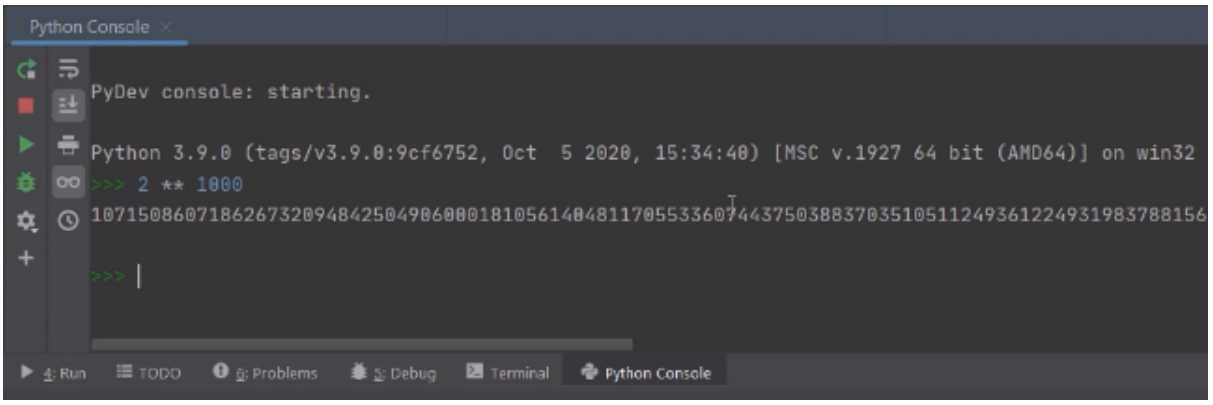


Во всплывающем окне, в разделе Condition, нужно указать условие. Теперь, если код достигнет этого места, будет выполнена проверка этого условия. Если условие выполняется, код будет остановлен, иначе код продолжит своё выполнение так, будто этой точки тут нет.

2. После установки точки нужно будет запустить дебаггер — сделать это можно, нажав на зеленого жучка в правой верхней части окна с кодом:



Ещё один инструмент в PyCharm — интерактивный режим выполнения кода (или консоль). Вызвать консоль можно, нажав на кнопку Python Console в окошке ниже. Выглядит она так:



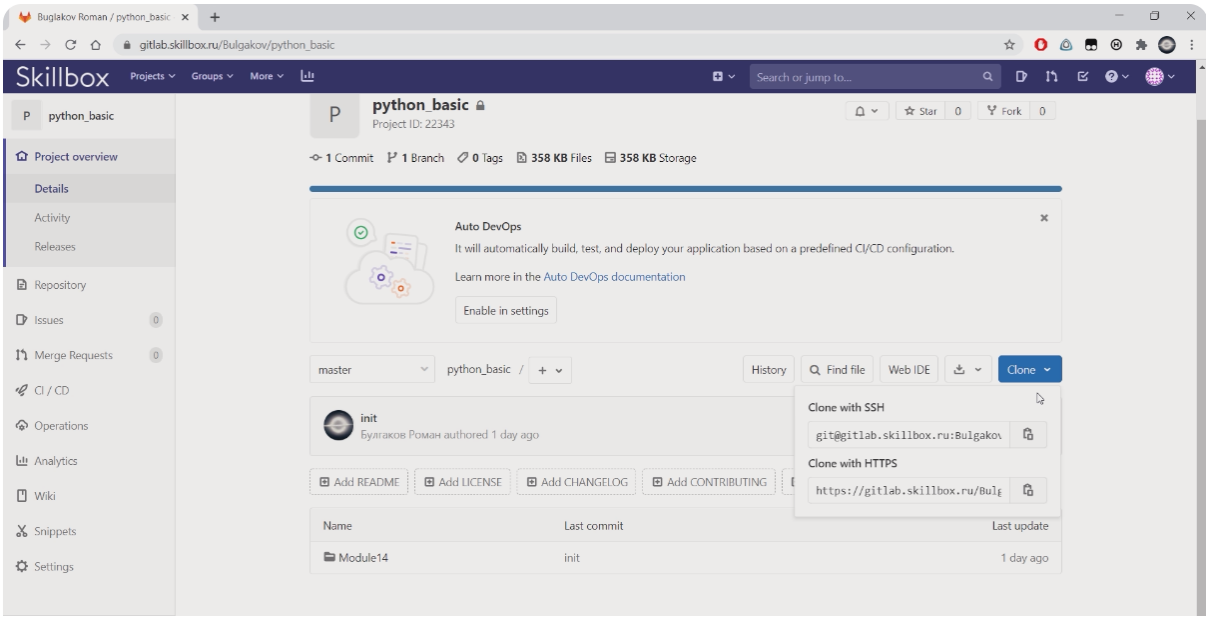
Зелёные стрелки указывают на то, что консоль ожидает ввод какого-либо Python-кода. После ввода кода и нажатия клавиши Enter код будет тут же выполнен. Этот режим подходит для быстрой проверки каких-то небольших частей кода.

GitLab
GitLab — это удалённый сервер, на котором может храниться код ваших программ. Его назначением является возможность коллективной работы над программой и запись отдельных версий/состояний кода.

Так, работая над кодом, можно сохранить его текущее состояние и добавить к нему какую-нибудь подпись, например Version 1.1. После сохранения и отправки версии кода на сервер другой программист может взять вашу версию программы 1.1, доработать её у себя на компьютере и добавить на сервер обновлённую версию 1.2. После этого при желании вы можете взять его версию кода, опять обновить её и добавить следующую версию. Таким образом можно продолжать работу над общим проектом.

Кроме этого, ничто не мешает вести разработку в одиночку. В этом случае **GitLab** можно будет использовать просто для того, чтобы хранить ранние версии кода. Если вдруг вы напишите новую версию кода, а она не будет работать, вы всегда сможете обратиться к серверу и взять более раннюю версию, которая ещё работала.

Сам сайт **GitLab** выглядит так:



Слева — перечень разделов (на данном этапе нам понадобится только раздел Project overwiew — Details, но вы можете самостоятельно исследовать и другие разделы).

В этом окне видим текущее состояние проекта: его название, количество коммитов (сохранений кода), количество ветвей, размер файлов, последний коммит (например, на скриншоте название коммита — init, автор — Булгаков Роман; это значит, что Булгаков Роман сохранил свою версию кода и отправил её на сервер) и перечень файлов и директорий в проекте (сейчас у нас в проекте только директория Module14).

Как работать с репозиторием (проектом)?

1. Нужно установить связь между удалённой версией проекта (которая хранится на сервере) и локальной (которая хранится у вас на компьютере). Для этого вам необходимо получить ссылку на удалённый репозиторий (кнопка Clone — ссылка в разделе Clone with HTTPS) и вставить эту ссылку в PyCharm (выбрать вкладку Git — Clone, подробный перечень шагов вы можете найти в лекции «Работа с Gitlab. Сдача домашнего задания»).
2. После того, как связь будет создана, вы получите доступ к операциям Git-а (системы контроля версий). Самые популярные и нужные нам — commit/push/pull:
    - commit — сохранение текущей версии кода на вашем компьютере;
    - push — отправка всех коммитов (сохранений) на удалённый сервер;
    - pull — загрузка всех коммитов с удалённого сервера.
3. После установки связи работа будет заключаться в следующей последовательности действий:
    - pull — вы получаете данные с сервера;
    - изменяете состояние репозитория (создаёте новые файлы, изменяете старые файлы, добавляете свой код);
    - сохраняете изменённое состояние — делаете commit;

- push — пушите (отправляете) созданные коммиты на сервер.

Не допускай следующих ошибок!

При работе с командной строкой не забывайте проверять путь, в котором открыта ваша командная строка в текущий момент. Сопоставляйте при необходимости этот путь с положением файлов, к которым вы обращаетесь.

При работе с дебаггером не забывайте создавать точки остановы, помните, что их можно создать больше, чем одну.

При работе с GitLab старайтесь выработать привычку выполнять PULL перед каждым началом работы и выполнять PUSH после завершения работы. Если в одном репозитории работает несколько человек, то отсутствие такой привычки может привести к конфликтам версий (и даже такая привычка не защищает на 100% от конфликтов версий, но с ними тоже можно работать — руководство по решению таких конфликтов в PyCharm можно найти тут).