

## 3

# IDE PYCHARM

### 3.1 Установка Python

## 3.2 Установка PyCharm

### 3.3 Создание проекта

### 3.4 Интерфейс проекта

### 3.5 Запуск и отладка программы

### 3.6 Справочные системы Python и PyCharm

### 3.7 Терминология

### 3.8 Контрольные вопросы и упражнения



# 3

## IDE PyCharm

### 3.1 Установка Python

Разработка программ на языке Python начинается с установки интерпретатора и стандартной библиотеки Python. Существует две параллельные ветки Python – 2.x и 3.x, которые доступны для различных платформ (Windows, Linux/Unix, Mac OS X и др.) на официальном сайте <https://www.python.org/downloads/>. Далее будем рассматривать Python версии 3.7.

**Примечание:**

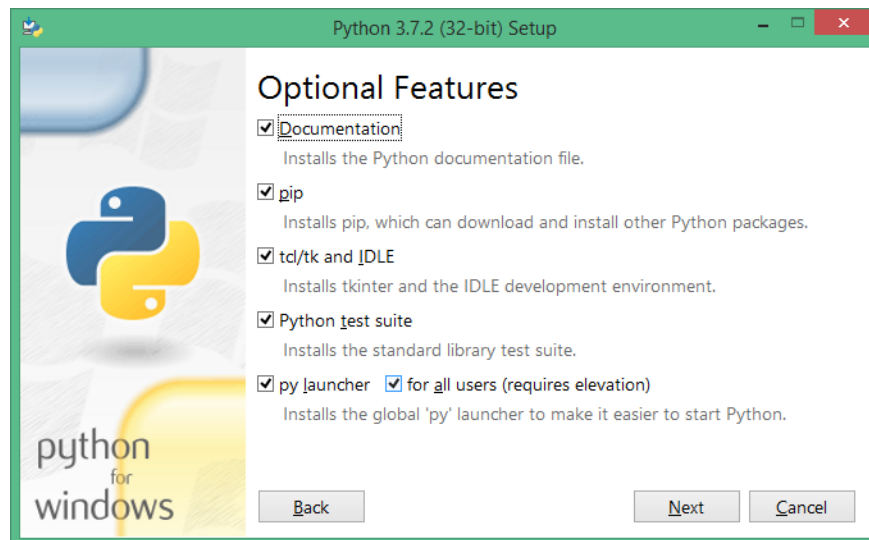
- ветка 3.x обратно несовместима с веткой 2.x;
- поддержка ветки 2.x продлена до 2020 г.

В дистрибутивы Linux, Mac OS включен Python версии 2.x и 3.x (по умолчанию используется Python 2.x). Для Windows доступны две версии инсталлятора: для 32-х битной и 64-х битной систем. Установка Python в Windows проходит в несколько шагов:

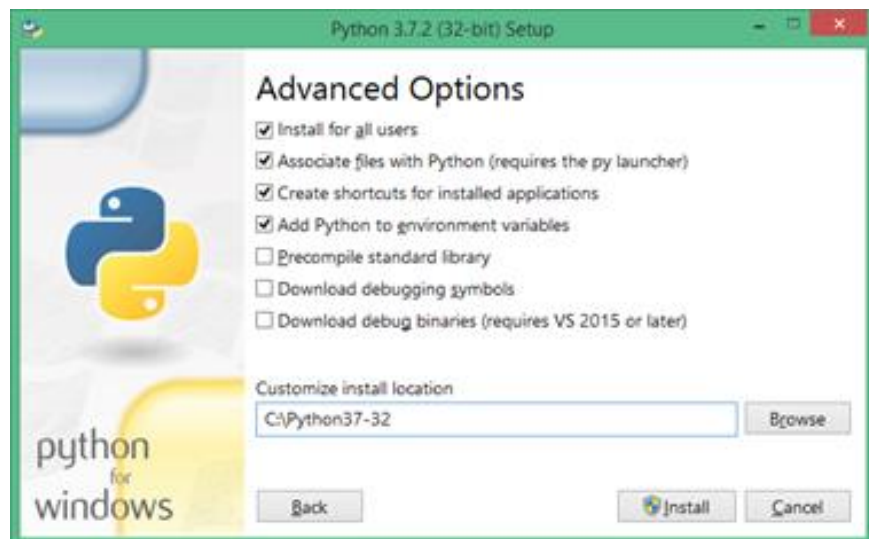
1. Запуск установщика.
2. Выбор типа установки: по умолчанию или настраиваемая. В нашем случае – настраиваемая.
3. Выбор компонент: файл с документацией, менеджер пакетов pip, графическая библиотека tkinter и интегрированная среда разработки и обучения IDLE, стандартный набор тестов, Python Launcher для запуска программ двойным щелчком мыши (рис. 3.1а). В нашем случае соглашаемся со всеми компонентами и нажимаем кнопку Next.
4. Выбор дополнительных настроек (рис. 3.1б). В нашем случае выбираем режим установки – для всех пользователей, создаем ярлыки для установленных приложений, устанавливаем ассоциацию с py-файлами, добавляем Python в переменные среды операционной системы, выбираем путь установки и нажимаем кнопку Install.
5. Завершение процесса установки (рис 3.1в). Проверить успешность установки можно с помощью команды `python -V` (возвращает версию установленного Python) в командной строке Windows (Win+R → cmd). В нашем случае:

```
C:\>python -V
```

```
Python 3.7.2
```



a)



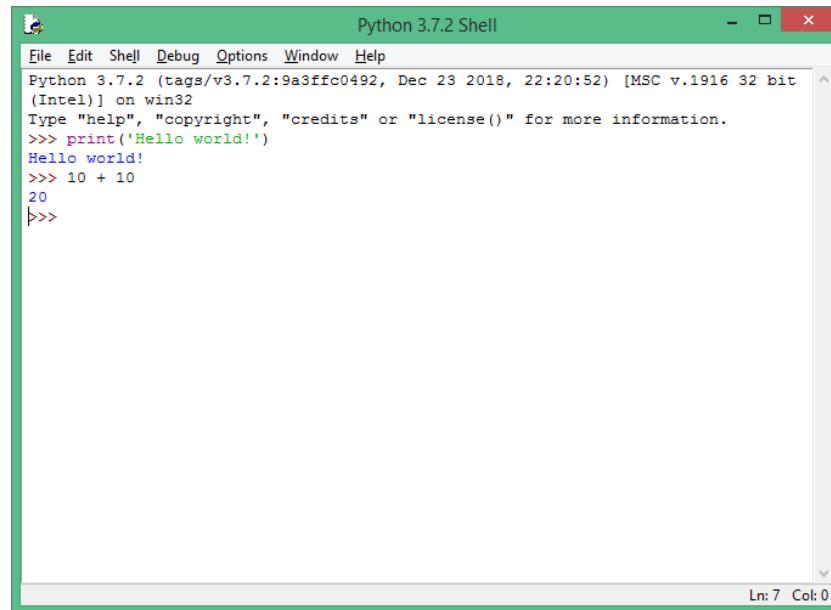
б)



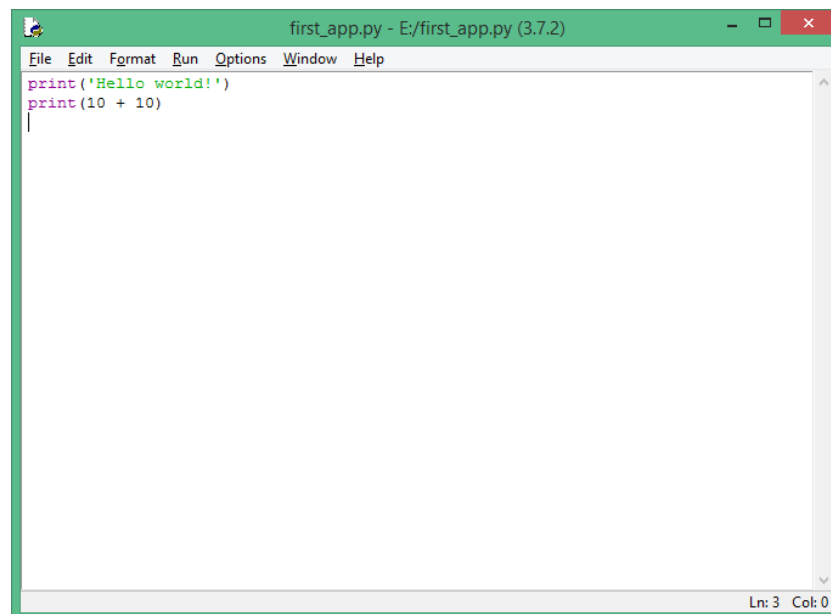
в)

Рис. 3.1. Работа с инсталлятором Python 3.7.2  
(<https://www.python.org/ftp/python/3.7.2/python-3.7.2.exe>)

После установки Python можно переходить к знакомству с языком и созданию первой программы – для этого необходимо запустить IDLE (Пуск → IDLE (Python 3.7)). IDLE (названа в честь Eric Idle из Монти Пайтон) покрывает стандартную схему подготовки программы на Python – редактирование, трансляцию и отладку. IDLE работает в двух режимах (рис. 3.2): интерактивный режим (IDLE Shell) и режим редактирования (IDLE Editor).



а)



б)

Рис. 3.2. Режимы работы IDLE

а) – IDLE Shell; б) – IDLE Editor

IDLE Shell (рис. 3.2а) выполняет команды построчно. Ввод команд возможен после приглашения >>>, например, для вывода текстового сообщения 'Hello world!' достаточно написать print('Hello world!') и нажать на Enter.

```
>>> print('Hello world!')

Hello world!
```

IDLE Shell чаще всего используется в качестве калькулятора, для изучения особенностей Python и отладки небольших фрагментов кода.

IDLE Editor (рис. 3.26) позволяет редактировать исходный код программы и результат сохранять в текстовый файл с расширением \*.py. Для перехода в режим редактирования необходимо в окне **Shell** выбрать пункт меню **File → New File (Ctrl+N)**, для возвращения в интерактивный режим – пункт меню **Run → Python Shell**. Для запуска программы необходимо сохранить исходный код программы и выбрать пункт меню **Run → Run Module (F5)**. IDLE Editor поддерживает многооконность, работу с буфером обмена операционной системы, подсветку синтаксиса, частичное авто-форматирование и авто-отступы, навигацию по коду.

Несмотря на свою простоту в использовании, IDLE больше подходит для ознакомления с Python и значительно проигрывает современным интегрированным средам разработки.

## 3.2 Установка PyCharm

PyCharm – интеллектуальная IDE с полным набор инструментов для эффективной разработки на Python. Существует два варианта поставки PyCharm – Community и Professional, которые доступны для различных платформ Windows, Linux и Mac OS на официальном сайте <https://www.jetbrains.com/pycharm/download>. В табл. 3.1 приведены основные возможности PyCharm.

Таблица 3.1. Возможности PyCharm

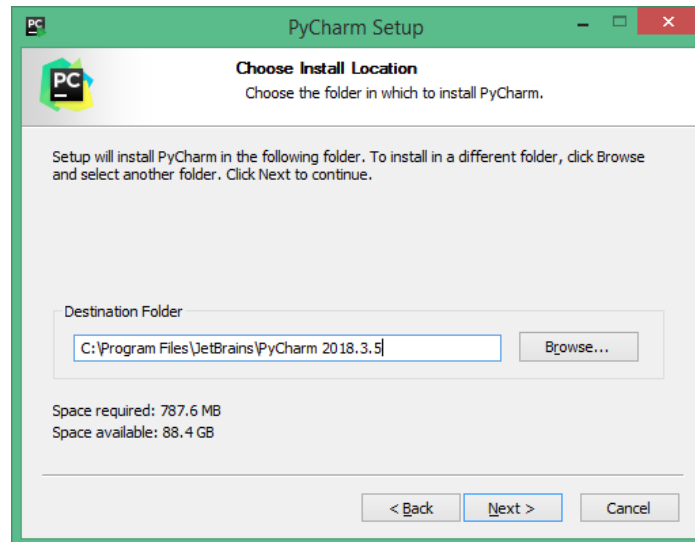
Характеристика	PyCharm Community	PyCharm Professional
интеллектуальный редактор	+	+
отладчик с графическим интерфейсом и инструменты тестирования	+	+
навигация (англ. navigation) и рефакторинг (англ. refactorings)	+	+
инспекция кода (англ. code inspections)	+	+
поддержка системы управления версиями (англ. Version Control System, VCS)	+	+

научные пакеты (интегрируется с IPython Notebook, поддерживает Anaconda, Pandas, Numpy, Matplotlib и др.)	-	+
поддержка веб-разработки (JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js и др.)	-	+
поддержка веб-фреймворков (Django, Flask, Google App Engine, Pyramid и web2py)	-	+
Python профилировщик (англ. profiler)	-	+
возможности удаленной разработки	-	+
поддержка работы с базами данных и SQL	-	+

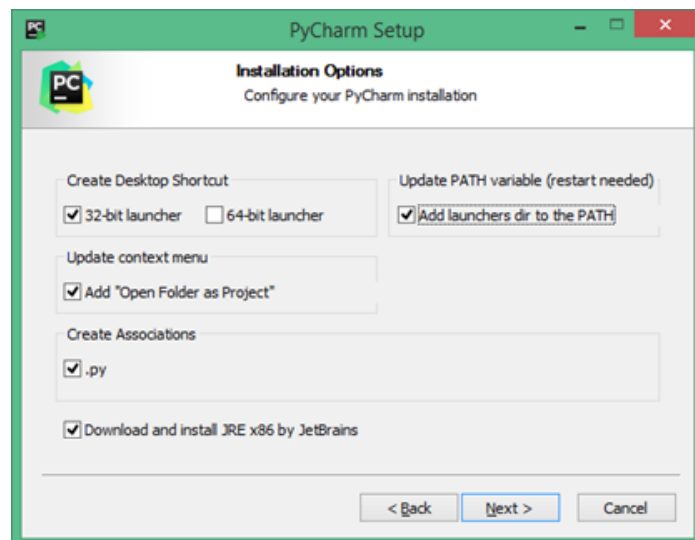
Рассмотрим основные шаги установки PyCharm Professional v. 2018.3.5 для Windows:

1. Запуск установщика. Мастер установки порекомендует закрыть все приложения и нажать кнопку Next для продолжения установки.
2. Выбор пути установки (рис. 3.3а). По умолчанию мастер выбирает стандартную директорию Program Files.
3. Выбор дополнительных настроек (рис. 3.3б): создаем ярлык, устанавливаем ассоциацию с py-файлами, добавляем ярлыки в переменную среды PATH, добавляем в контекстное меню Open Folder As Project, устанавливаем JRE.
4. Завершение процесса установки (рис. 3.3в). Для корректной работы PyCharm мастер предложит перезагрузить компьютер.
5. Импорт настроек. При первом запуске либо после обновления PyCharm будет открыто диалоговое окно, в котором будет предложено импортировать настройки PyCharm, если таковы имеются.
6. Выбор темы. При первом запуске либо после обновления PyCharm будет предложено выбрать тему пользовательского интерфейса (Darcula / Light, рис. 3.4).
7. Выбор и установка дополнительных плагинов. На последнем шаге будет предложено загрузить и установить дополнительные плагины из хранилища плагинов PyCharm – IdeaVim, Markdown, BashSupport, R Language Support (рис. 3.5).

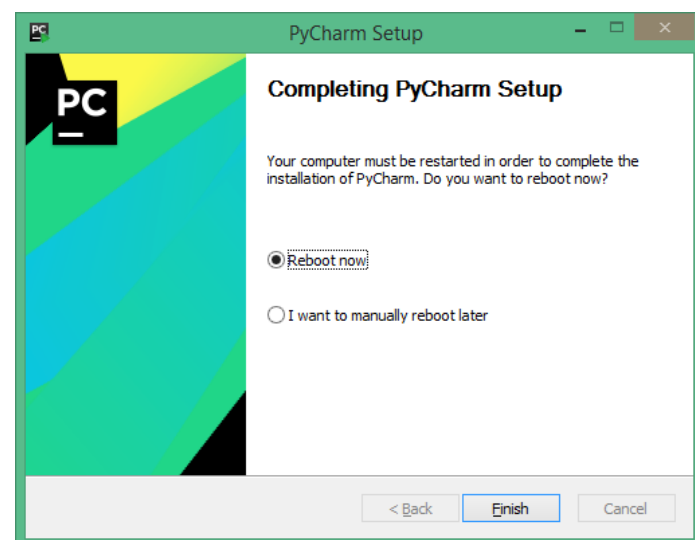
Инструкция по установке PyCharm для Linux и Mac OS доступна на официальном сайте в разделе Meet PyCharm → Requirements, Installation and Launching (<https://www.jetbrains.com/help/pycharm/install-and-set-up-pycharm.html>).



a)



б)



в)

Рис. 3.3. Работа с инсталлятором PyCharm  
(<https://www.jetbrains.com/pycharm/download/>)

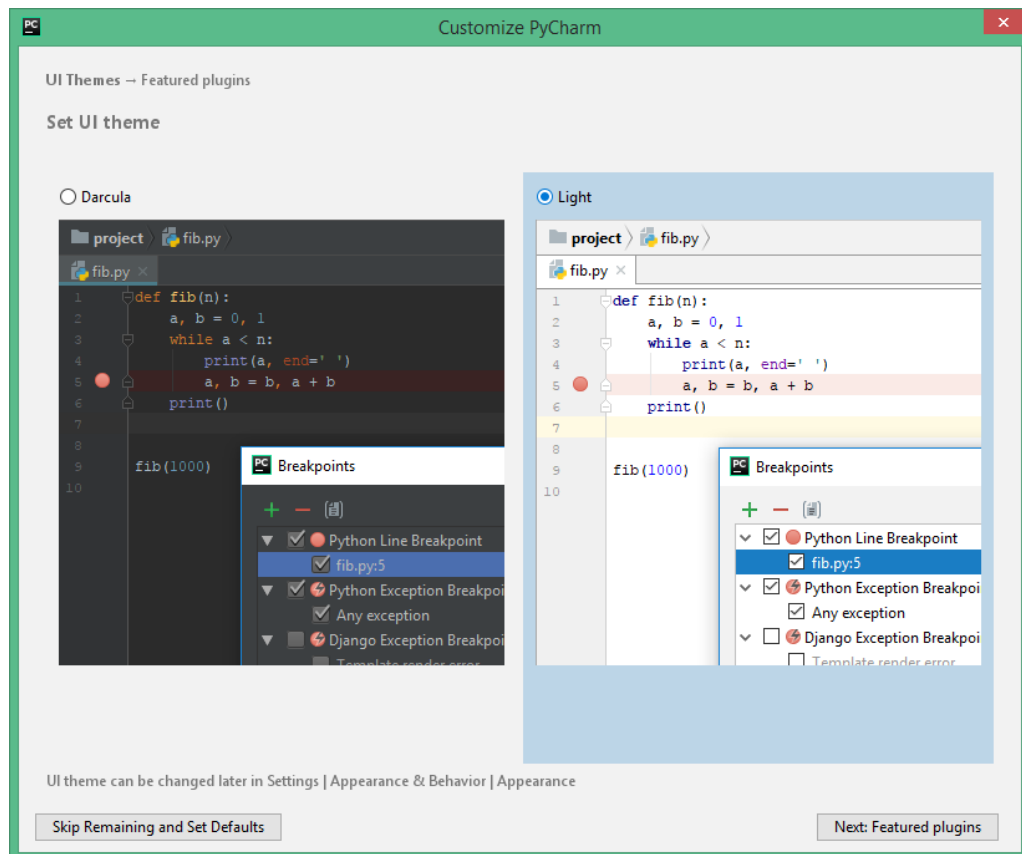


Рис. 3.4. Кастомизация PyCharm. Выбор темы

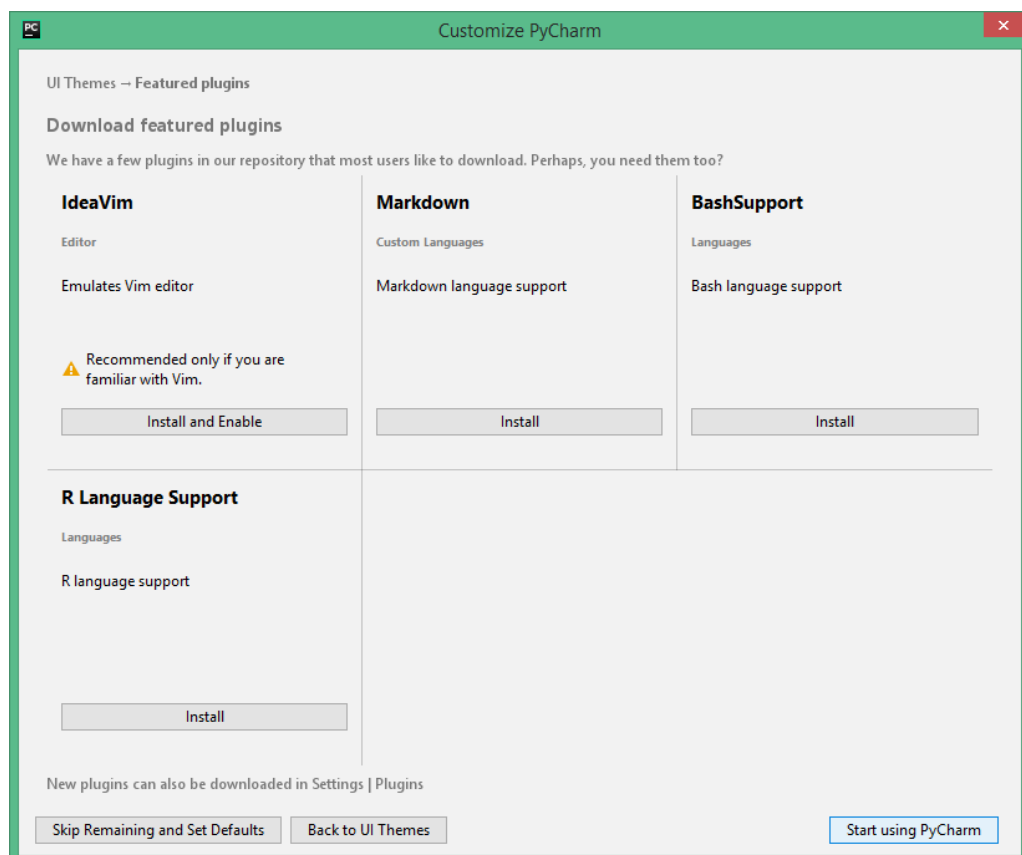


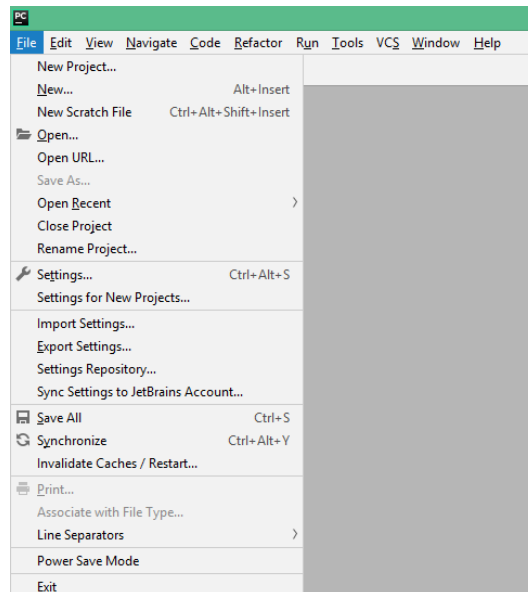
Рис. 3.5. Кастомизация PyCharm. Выбор дополнительных плагинов



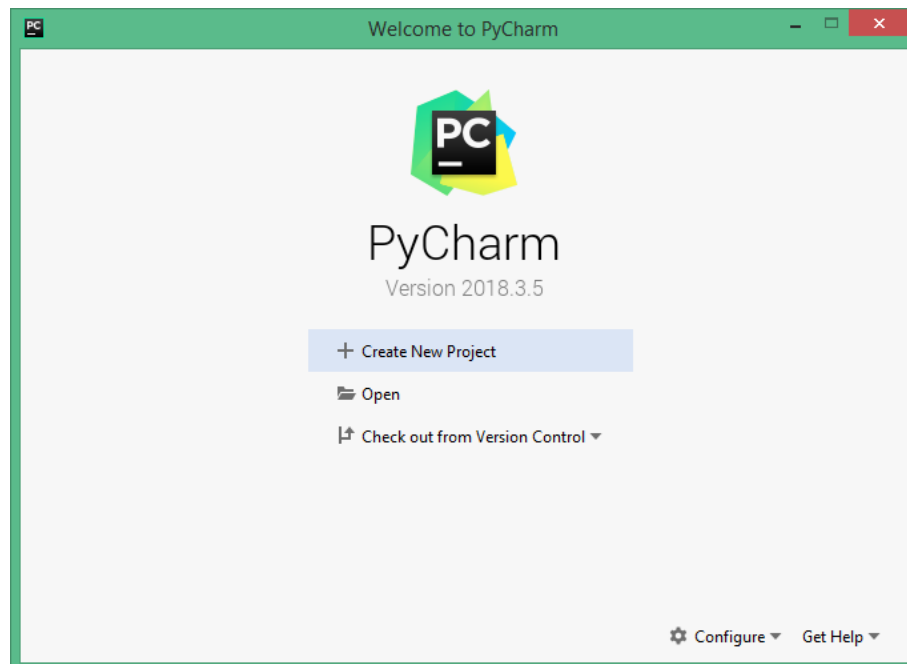
### 3.3 Создание проекта

Разработка программ в PyCharm выполняется в контексте проекта (англ. project), который объединяет файлы с исходным кодом программы и специальные файлы с настройками проекта, предоставляет возможности для редактирования, рефакторинга и т. д. Создание проекта в PyCharm выполняется в несколько этапов:

1. Выбрать **File → New Project** или **Welcome Window → Create New Project** (рис. 3.6).



a)



б)

Рис. 3.6. Создание нового проекта в PyCharm

а) – **File → New Project**; б) – **Welcome Window → Create New Project**

2. В диалоговом окне **New Project** в секции **Project type** выбрать тип проекта. Для создания «чистого» проекта необходимо выбрать тип **Pure Python** и далее указать местоположение проекта, выбрать существующий интерпретатор или создать виртуальное окружение (рис. 3.7). PyCharm позволяет использовать `virtualenv` tool для создания изолированного виртуального окружения для конкретного проекта, что дает возможность управлять настройками интерпретатора независимо от других проектов.

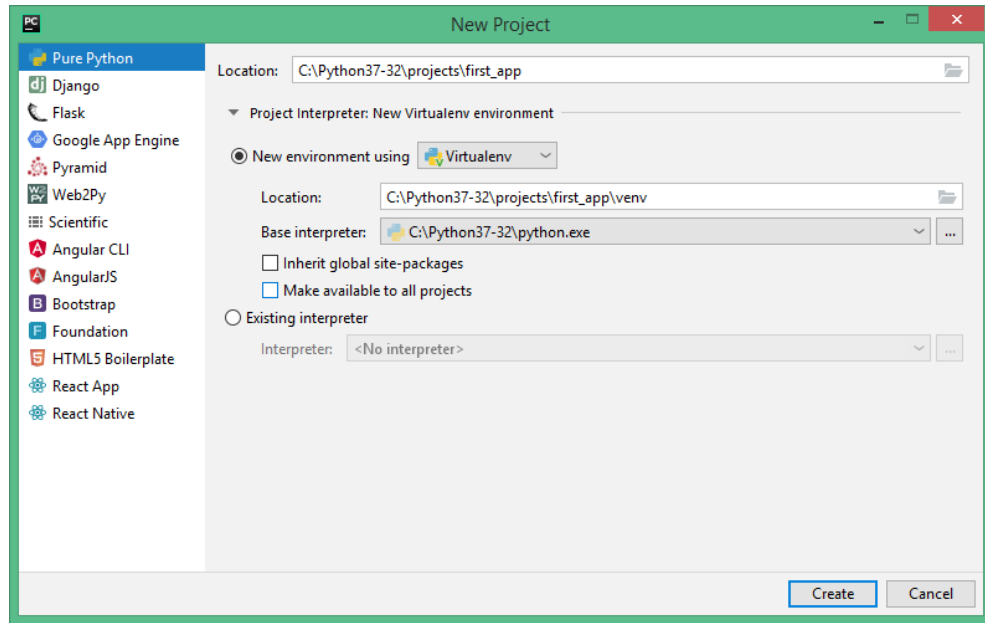


Рис. 3.7. Настройка параметров проекта в PyCharm

Проект Pure Python имеет следующую структуру:

- a) каталог **.idea** (не отображается в обозревателе проекта – **Project view**) со следующими файлами:
    - **.iml** – файл, описывающий структуру проекта.
    - **workspace.xml** – файл, содержащий настройки рабочего пространства пользователя.
    - **xml-файлы**, содержащие наборы настроек (например, **vcs.xml** – настройки VCS, **encodings.xml** – тип кодировки и т. д.).
  - b) каталог **.venv** с файлами виртуального окружения (в случае выбора **New environment** в диалоговом окне **New Project**).
  - c) пользовательские каталоги, пакеты и **py-файлы** с исходным кодом программы.
3. В **Project view** правой кнопкой мыши выбрать имя проекта и далее в контекстном меню выбрать **New → Python File**. В диалоговом окне в текстовом поле **Name** ввести имя файла и выбрать тип файла **Python** (рис. 3.8).

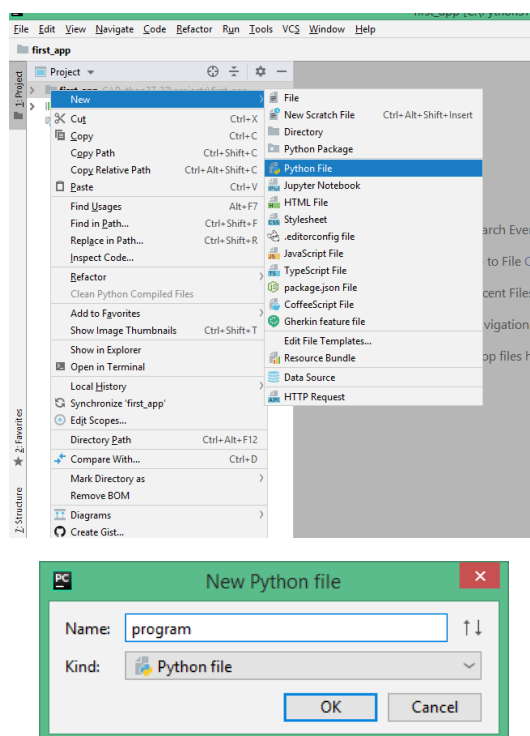


Рис. 3.8. Создание ру-файла в PyCharm

В главном окне PyCharm можно выделить пять основных элементов (рис. 3.9).

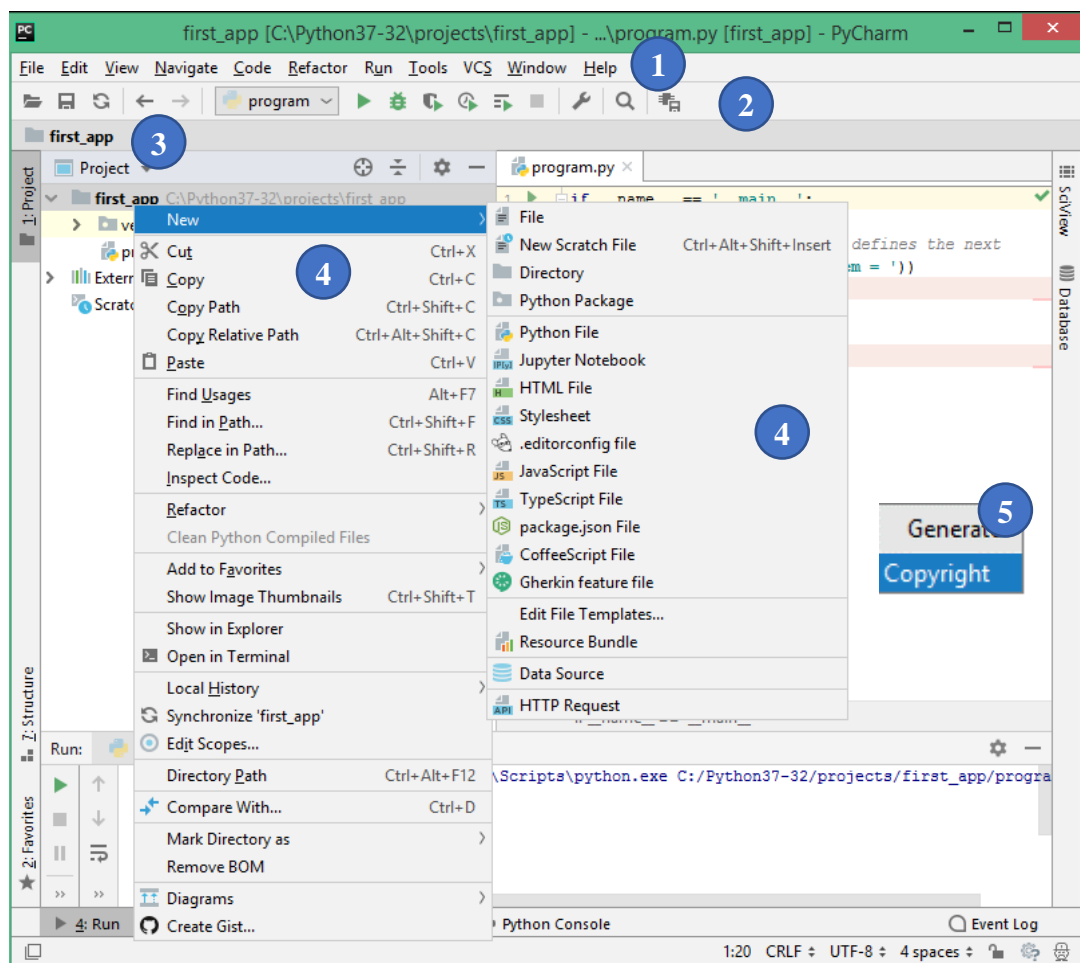


Рис. 3.9. Основные элементы главного окна PyCharm

1. **Main menu** – меню, которое содержит группы взаимосвязанных команд для управления процессом разработки ПО. В табл. 3.2 приведено краткое описание основных пунктов меню.

Таблица 3.2. Описание пунктов меню PyCharm

Пункт меню	Назначение
File	Содержит команды создания, открытия, закрытия и настройки проектов, а также выхода из PyCharm
Edit	Содержит команды редактирования программ
View	Содержит команды для работы с панелями инструментов PyCharm
Navigate	Содержит команды навигации по проекту (например, по классам, файлам и т.д.)
Code	Содержит команды для работы с исходным кодом программы – генерация, завершение и инспекция кода, поиск дублирующего кода
Refactor	Содержит команды для рефакторинга кода
Run	Содержит команды управления запуском и отладкой программы
Tools	Содержит команды по работе с дополнительными инструментами
VCS	Содержит команды по управлению версиями проекта
Window	Содержит команды управления окнами проекта
Help	Содержит команды для обращения к справочной системе PyCharm

Подробное описание подпунктов для перечисленных пунктов **Main menu** доступно в справочных материалах в разделе **Reference → Index of Menu Items**.

2. **Main toolbar** – панель инструментов, которая состоит из элементов управления, дублирующих основные команды **Main menu** для быстрого доступа. По умолчанию панель инструментов скрыта и для ее отображения необходимо выбрать **View → Toolbar**.
3. **Navigation bar** – навигационная панель, которая является альтернативой **Project view**.
4. **Context menus** – контекстные меню, которые вызываются пользователем при щелчке правой кнопкой мыши и представляют собой список команд, применимые к выбранному объекту (текущему контексту).
5. **Pop-up menus** – всплывающие меню, которые вызываются пользователем по **Alt+Insert** и представляют собой список команд, применимые к выбранному объекту (текущему контексту).

### 3.4 Интерфейс проекта

Интерфейс проекта в PyCharm разделен на несколько логических областей (рис. 3.10).

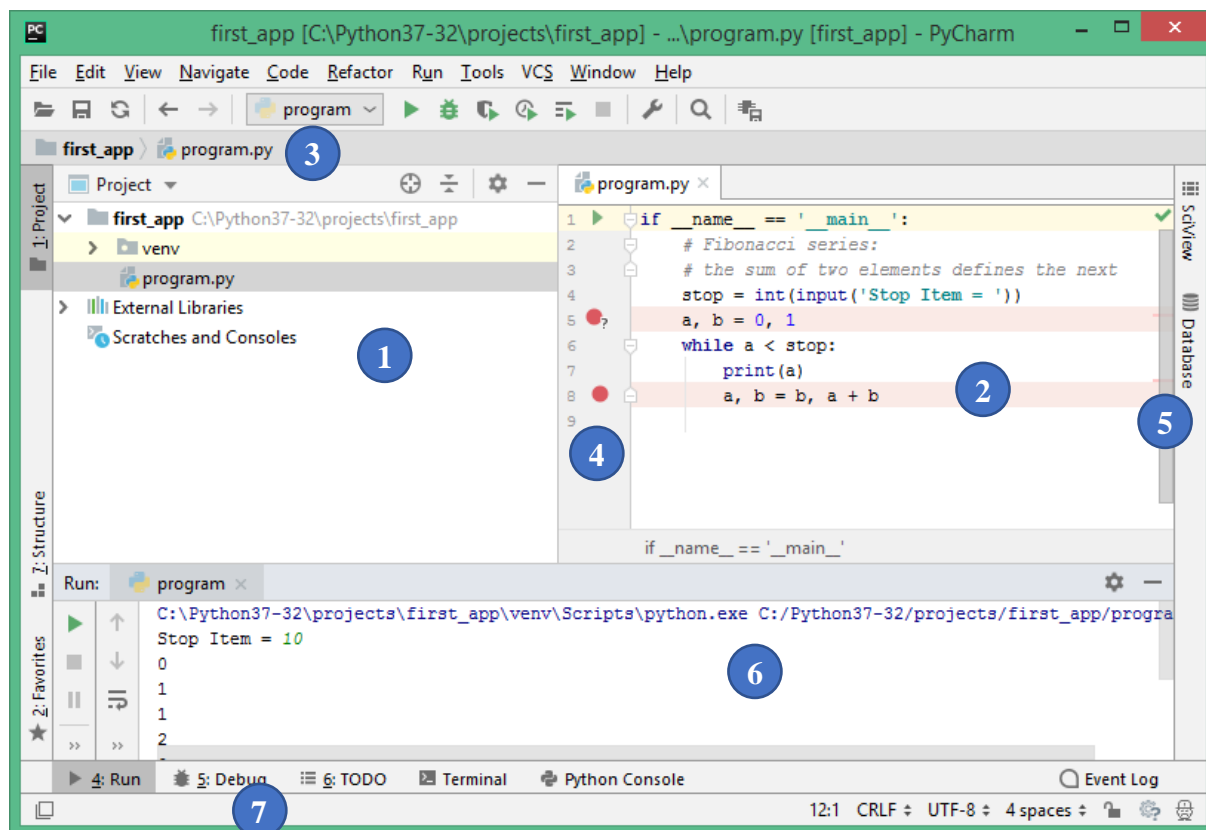


Рис. 3.10. Интерфейс проекта в PyCharm

1. **Project view** – обозреватель проекта, в котором отображаются файлы проекта.
2. **Editor** – редактор для создания и изменения исходного кода программы. Редактор кода поддерживает вкладки для быстрой навигации между открытыми файлами, подсветку синтаксиса, авто-форматирование, интеллектуальное завершение кода, свертывание блоков кода и многое другое.
3. **Navigation Bar** – навигационная панель, которая состоит из элементов управления для быстрого запуска, отладки и контроля версий программы.
4. **Left gutter** – столбец слева от **Editor**, который отображает установленные точки останова, номера строк кода, историю VCS, а также обеспечивает удобный способ навигации по иерархии кода.
5. **Right gutter** – столбец справа от **Editor**, который отображает результаты проверки качества исходного кода.
6. **Tool Windows** – прикрепленные снизу и по сторонам рабочей области панели инструментов, которые обеспечивают доступ к типичным задачам, например, управление проектом, поиск и навигация по исходному коду программы и т.д.
7. **Status Bar** – статусная строка, в которой отображаются состояние проекта и PyCharm, различные предупреждения и информационные сообщения.

Подробное описание элементов интерфейса проекта в PyCharm доступно в справочных материалах в разделе **Meet PyCharm → Quick Start Guide**. Для настройки интерфейса проекта в соответствии с потребностями пользователя необходимо выбрать **File → Settings**.

### 3.5 Запуск и отладка программы

Запустить программу в PyCharm можно несколькими способами (▶ – значок запуска программы):

1. Выбрать пункт меню **Run → Run...** либо воспользоваться комбинацией клавиш **Shift+Alt+F10** и далее в контекстном меню выбрать необходимую конфигурацию запуска (рис. 3.11а).
2. Нажать на значок запуска, расположенный на **Left gutter** (доступен при наличии в программе конструкции `if __name__ == '__main__':`) (рис. 3.11б).
3. Выбрать необходимую конфигурацию запуска в контекстном меню **Editor** (рис. 3.11в).
4. В **Project view** выбрать необходимый ру-файл и в контекстном меню выбрать **Run <name>** (рис. 3.11г).
5. Выбрать необходимую конфигурацию запуска на **Main toolbar** (рис. 3.11д).

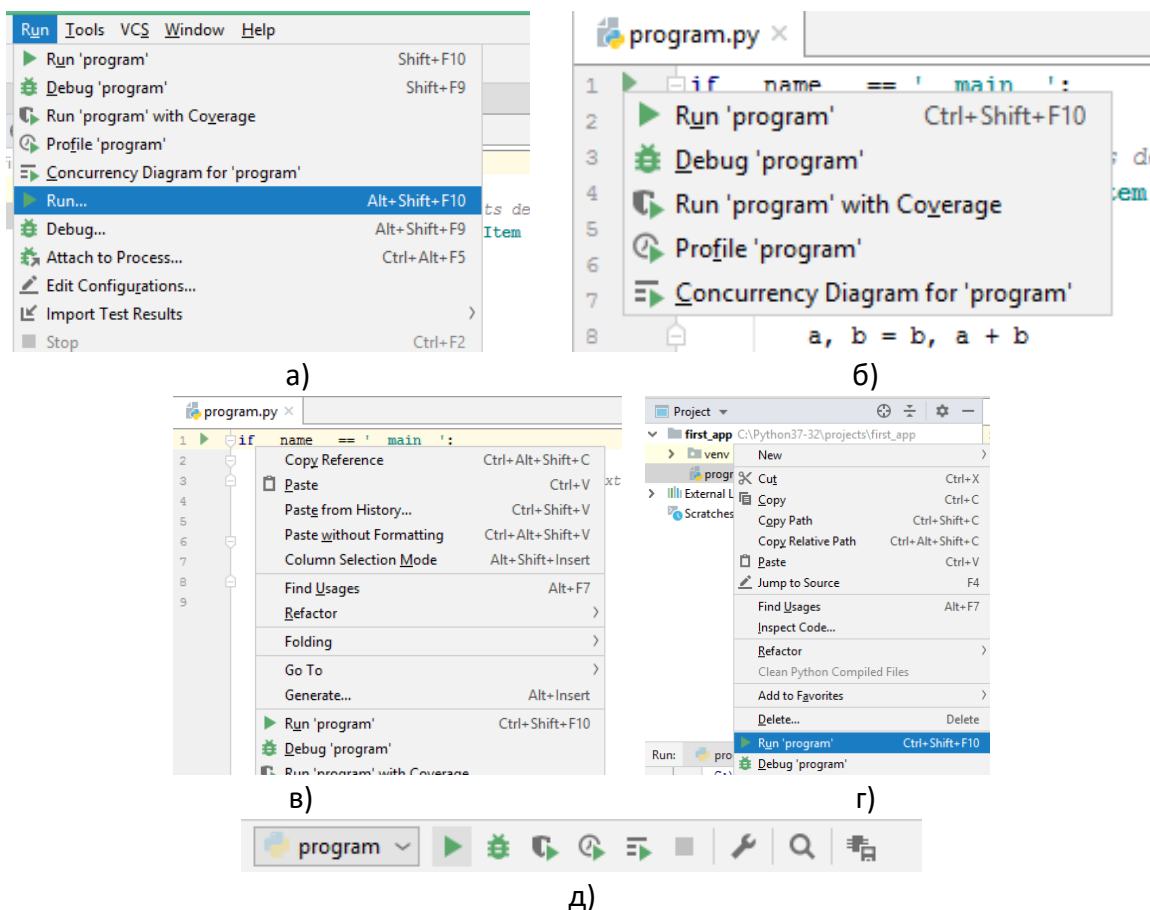


Рис. 3.11. Запуск программ в PyCharm

Результаты выполнения программы будут отображены в панели инструментов **Run** (рис. 3.12).



Рис. 3.12. Панель инструментов **Run** в PyCharm

PyCharm при запуске программы автоматически создает временную конфигурацию запуска / отладки (временная конфигурация может быть сохранена как постоянная). Для просмотра параметров конфигурации необходимо выбрать пункт меню **Run → Edit Configurations...** либо выбрать **Edit Configurations** на **Main toolbar**, после чего в диалоговом окне выбрать необходимый py-файл проекта (рис. 3.13).

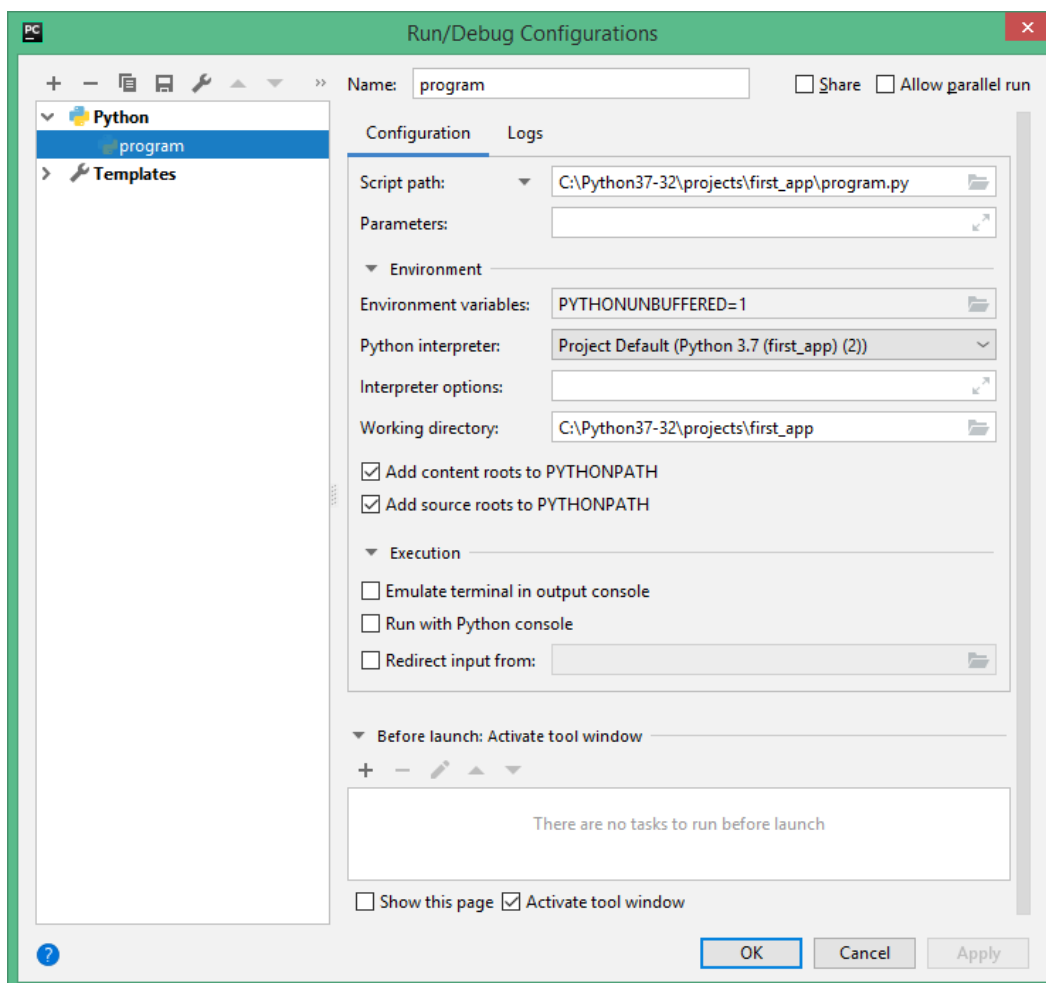


Рис. 3.13. Параметры конфигурацию запуска / отладки программы

Диалоговое окно **Run/Debug Configurations** содержит две вкладки: **Configuration** и **Logs**. Вкладка **Configuration** условно разделена на 4 секции:

1. Путь и параметры запускаемой программы (параметры: **Script path, Parameters**).
2. Настройки окружения и интерпретатора (параметры: **Environment variables, Python interpreter, Interpreter options, Working directory, Add content roots to PYTHONPATH, Add source roots to PYTHONPATH**).
3. Настройки выполнения программы (параметры: **Emulate terminal in output console, Run with Python console, Redirect input**).
4. Определение дополнительных инструментов или сценариев, которые будут выполнены до запуска программы.

Вкладка **Logs** позволяет выбрать log-файлы (создаются во время выполнения или отладки программы) для отображения в консоли, т. е. на выделенных вкладках панелей инструментов **Run** и **Debug**.

При выполнении программы может быть обнаружено, что результат работы программы неверный или программа имеет неожиданное поведение. Первичным инструментом для поиска логических ошибок в IDE является отладчик, предоставляющий программисту набор команд для трассировки (пошагового выполнения) и контроля состояния программы. Первым этапом при отладке программы является расстановка точек останова (англ. breakpoint) – специальные метки в исходном коде программы, на которых происходит остановка выполнения программы. При остановке программы отладчик позволяет программисту исследовать программу на корректность ее работы (например, просмотреть состояние переменных, вычислить значение выражения). Расстановка точек останова (● – значок точки останова) осуществляется с помощью нажатия левой кнопки мыши на Left gutter на нужных строках исходного кода программы (рис. 3.14).

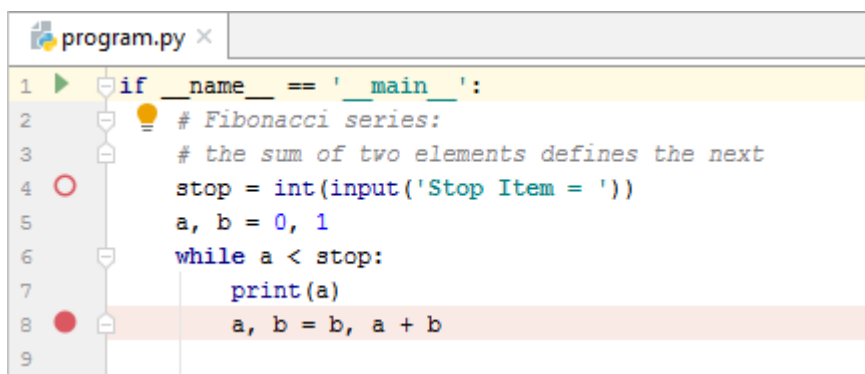


Рис. 3.14. Установка точек останова в PyCharm

Управление точками останова осуществляется с помощью диалогового окна **Breakpoints** (рис. 3.15), для вызова которого необходимо выбрать пункт меню **Run → View Breakpoints...** (Ctrl+Shift+F8).



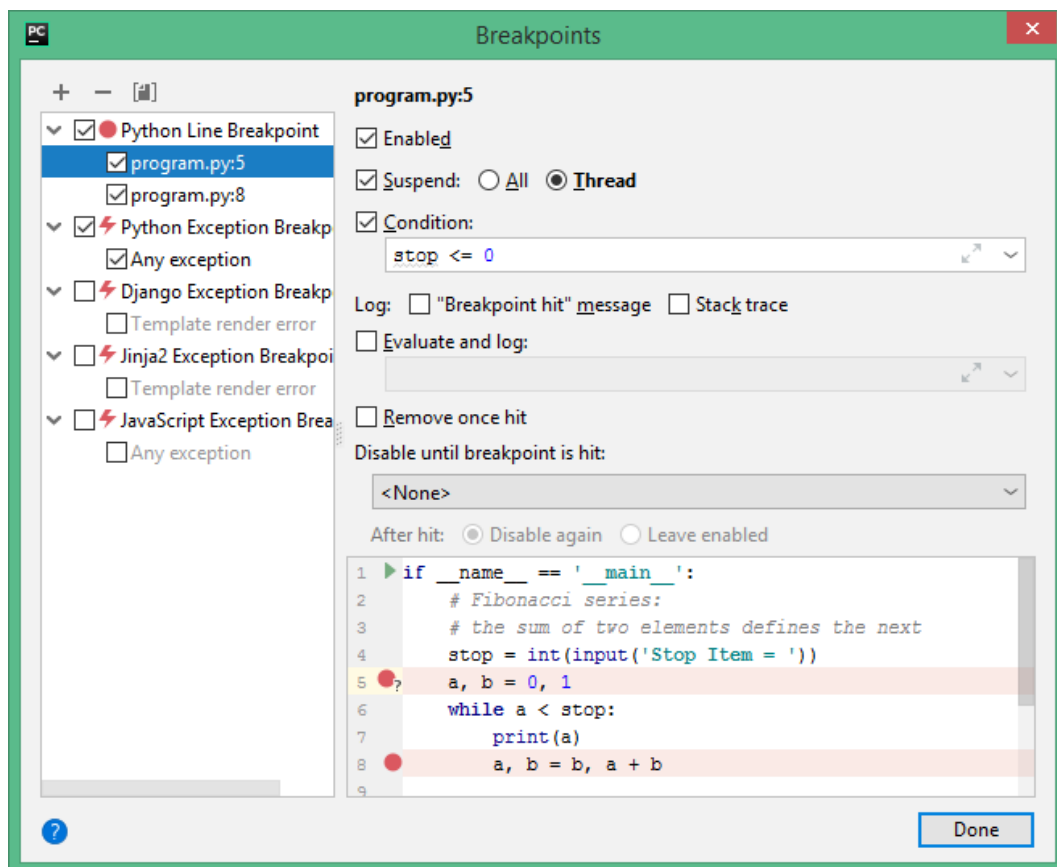


Рис. 3.15. Управление точками останова в PyCharm


В диалоговом окне **Breakpoints** справа отображаются установленные точки останова с указанием соответствующих номеров строк в исходном коде программы. После выбора точки останова будут доступны параметры управления точкой. Основные параметры: отключение точки останова, определение условия, при котором точка останова сработает, вычисление выражения для логирования.

Запустить программу в режиме отладки в PyCharm можно несколькими способами (🐞 – значок отладки программы):

1. Выбрать пункт меню **Run → Debug...** либо воспользоваться комбинацией клавиш **Alt+Shift+F9** и далее в контекстном меню выбрать необходимую конфигурацию отладки (рис. 3.16а).
2. Выбрать необходимую конфигурацию отладки на **Main toolbar** (рис. 3.16г).
3. В **Project view** выбрать необходимый ру-файл и в контекстном меню выбрать **Debug <name>** (рис. 3.16б).
4. Выбрать необходимую конфигурацию запуска в контекстном меню **Editor** (рис. 3.16в).

Для управления работой программы в режиме отладки будет открыта панель инструментов **Debug** (рис. 3.17). Выполнение программы будет остановлено на первой точке останова, после чего можно будет перейти к анализу исходного кода программы. Панель инструментов **Debug** разделена на три секции:

1. **Frames** – позволяет получить доступ к списку потоков программы.

2. **Variables** – позволяет просматривать состояние объектов программы.
3. **Watches** – позволяет просматривать состояния объектов программы, вычислять значения выражений в контексте текущего стекового кадра (англ. stack frame). Секция **Watches** по умолчанию скрыта. Для ее активации необходимо на панели инструментов секции **Variables** нажать на кнопку .

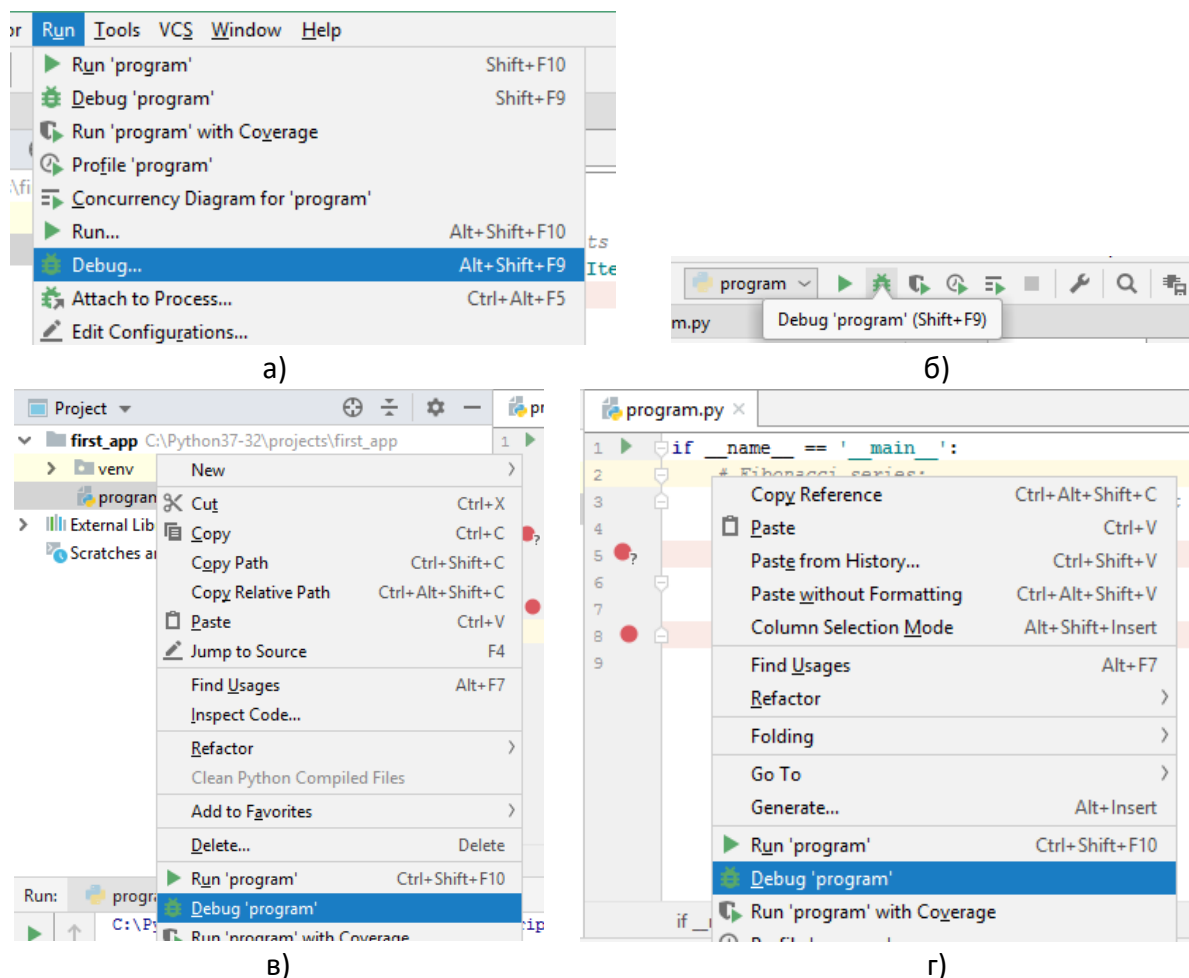
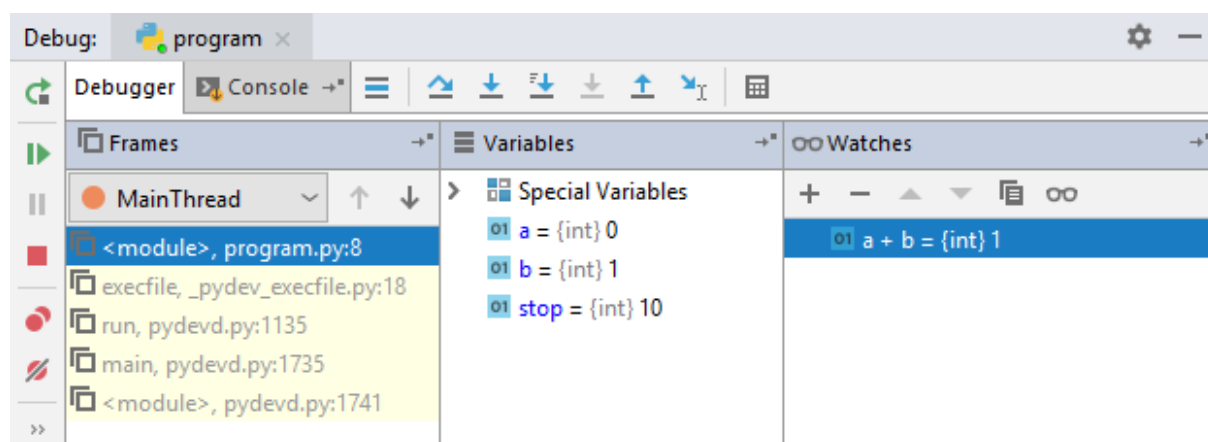


Рис. 3.16. Запуск отладчика в PyCharm

Рис. 3.17. Панель инструментов **Debug** в PyCharm

Управление выполнением программы в режиме отладки осуществляется с помощью команд, которые расположены на панели инструментов отладчика (рис. 3.18).

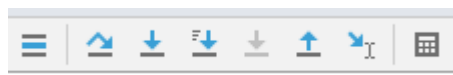


Рис. 3.18. Панель инструментов отладчика

В табл. 3.3 приведено описание команд отладчика.

Таблица 3.3. Команды отладчика в PyCharm

Команда	Горячие клавиши	Описание
	Alt+F10	Выделение в <b>Editor</b> текущей точки выполнения программы с соответствующим отображением стекового фрейма в секции Frames
	F8	Пошаговое выполнение программы без захода в вызываемые функции
	F7	Пошаговое выполнение программы с заходом в вызываемые функции
	Shift+Alt+F7	Пошаговое выполнение программы только собственного кода, т. е. библиотечные вызовы будут выполнены без останова
	Shift+F8	Выполнение функции и остановка на команде, следующей после вызова функции
	Alt+F9	Выполнение программы до места расположения курсора в <b>Editor</b>
	Alt+F8	Вычисление выражения

## 3.6 Справочные системы Python и PyCharm

Python имеет достаточно полную документацию, которая доступна на официальном сайте в разделе **Documentation** (<https://www.python.org/doc/>). Пользователи имеют возможность изучать особенности языка как в online режиме, так и скачать на локальный компьютер архив с документами определенного формата (PDF, HTML, Plain Text, EPUB). При работе с документацией необходимо также учитывать версию установленного Python. В документации имеются взаимные ссылки на связанные разделы, по языку программирования есть возможность получить не только формальное описание конструкций, но и примеры – фрагменты кода.

По PyCharm имеется также полная документация, которая доступна только в online режиме на официальном сайте в разделе **Docs & Demos** (<https://www.jetbrains.com/pycharm/documentation/>) либо при выборе пункта меню

**Help → Help.** Справка имеет взаимные ссылки на связанные разделы, по элементам IDE есть примеры экранных форм.

Справочные системы Python и PyCharm считаются достаточными для самостоятельного изучения как языка Python, так и IDE PyCharm.

## 3.7 Терминология

Python	навигация
интегрированная среда разработки	рефакторинг
IDLE	инспекция кода
PyCharm	система управления версиями
редактор	профилирование
интерпретатор	проект
виртуальное окружение	логическая ошибка
отладчик	точка останова

## 3.8 Контрольные вопросы и упражнения

### Контрольные вопросы:

1. Опишите назначение пунктов главного меню PyCharm.
2. Опишите элементы рабочего пространства в PyCharm.
3. Дайте определение понятию «проект». Перечислите основные составляющие проекта.
4. Опишите элементы интерфейса проекта в PyCharm.
5. Опишите инструменты в PyCharm по работе с исходным кодом программы.
6. Опишите инструменты в PyCharm по повышению качества исходного кода программы.
7. Опишите назначение функции `help( )` в Python.
8. Дайте определение понятию «точка останова». Как установить точку останова в PyCharm?
9. Как задать условие для точки останова в PyCharm?
10. Опишите назначение секций и команд отладчика в PyCharm.

### Вопросы для самостоятельной работы:

1. Навигация и рефакторинг. Инструменты навигации и рефакторинга в PyCharm.
2. Инспекция кода. Инструменты инспекции кода в PyCharm.
3. Профилирование кода. Инструменты профилирования кода в PyCharm.

4. Виртуальное окружение Python. Настройка интерпретатора Python в PyCharm.
5. Кастомизация интерфейса проекта в PyCharm.