Введение в язык программирования Python

Тема: «Введение в язык программирования Python, настройка среды разработки, работа с переменными».

Введение

Python – интерпретируемый, объектно-ориентированный высокоуровневый язык программирования с динамической семантикой. Встроенные высокоуровневые структуры данных в сочетании с динамической типизацией и связыванием делают язык привлекательным для быстрой разработки приложений. Кроме того, его можно использовать в качестве сценарного языка для связи программных компонентов. Синтаксис Python прост в изучении, в нем придается особое значение читаемости кода, а это сокращает затраты на сопровождение программных продуктов. Python поддерживает модули и пакеты, поощряя модульность и повторное использование кода. Интерпретатор Python и большая стандартная библиотека доступны бесплатно в виде исходных и исполняемых кодов для всех основных платформ и могут свободно распространяться.

Основные особенности языка программирования Python:

1. Скриптовый язык. Код программ определяется в виде скриптов.
2. Поддержка самых различных парадигм программирования, в том числе объектно-ориентированной и функциональной парадигм.
3. Интерпретация программ. Для работы со скриптами необходим интерпретатор, который запускает и выполняет скрипт.
4. Портативность и кроссплатформенность. Не имеет значения, какая у вас операционная система – Windows, Mac OS, Linux, вам достаточно написать скрипт, который будет запускаться на всех этих ОС при наличии интерпретатора.
5. Автоматическое управление памятью.
6. Динамическая типизация.

Выполнение программы

Выполнение программы на Python выглядит следующим образом. Сначала вы пишите в текстовом редакторе скрипт с набором выражений. Далее этот скрипт

передаётся на выполнение интерпретатору. Интерпретатор транслирует код в промежуточный байткод, а затем виртуальная машина переводит полученный байткод в набор инструкций, которые выполняются операционной системой.

Здесь стоит отметить, что хотя формально трансляция интерпретатором исходного кода в байткод и перевод байткода виртуальной машиной в набор машинных команд представляют два разных процесса, фактически они объединены в самом интерпретаторе.

Рис. 1. Выполнение программы на языке Python

Установка Python

Для создания программ на Python нам потребуется интерпретатор – https://www.python.org/downloads/

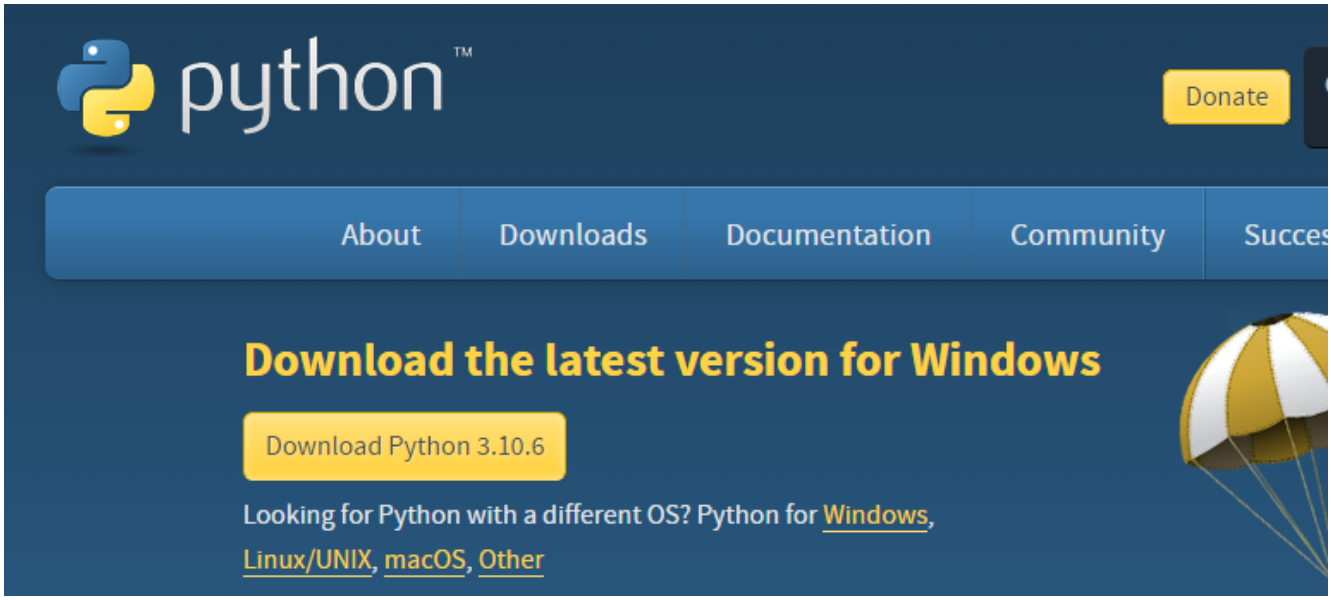

Рис. 2. Страница https://www.python.org/downloads/

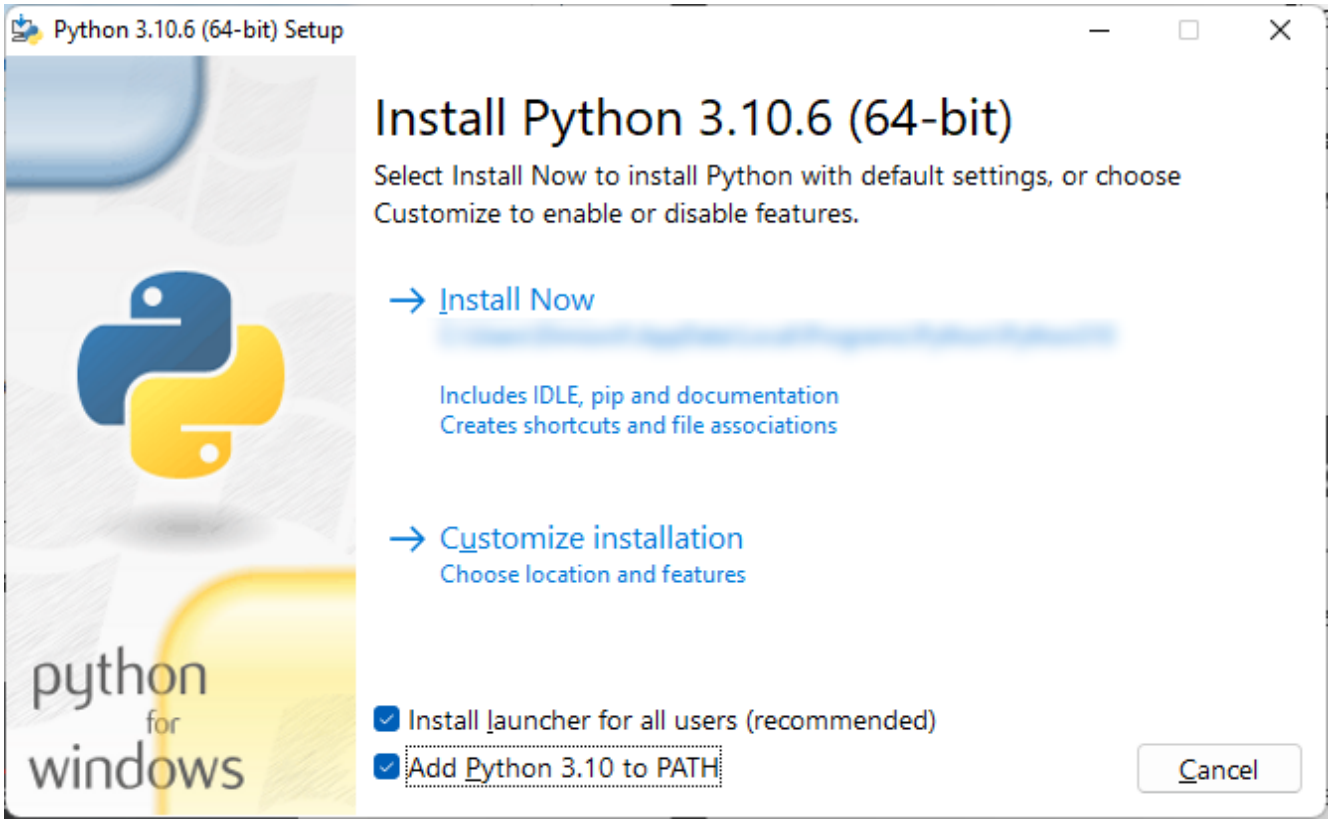

Рис. 3. Установка Python 3.10.6

Не забудьте отметить флажок «**Add Python 3.10 to PATH**», чтобы добавить путь к интерпретатору в переменные среды.

Создание файла программы

1. Создайте на диске **H** папку ***python*.**
2. В этой папке создайте новый текстовый документ, который назовите **hello.py**. *По умолчанию файлы с кодом на языке Python, как правило, имеют расширение .py.*
3. Откройте данный файл в любом текстовом редакторе и добавьте в него следующий код:

```
name = input("Введите имя: ")
print("Привет,", name)
```

Скрипт состоит из двух строк. Первая строка с помощью метода `input()` ожидает ввода пользователем своего имени. Введенное имя затем попадает в переменную `name`.

Вторая строка с помощью метода `print()` выводит приветствие вместе с введенным именем.

Теперь запустите командную строку/терминал и выполните команду:

```
python H:\python\hello.py
```

Программа выведет приглашение к вводу имени, а затем приветствие.

IDE PyCharm

IDE предоставляет нам текстовый редактор для набора кода, но в отличие от стандартных текстовых редакторов, IDE также обеспечивает полноценную подсветку синтаксиса, автодополнение, интеллектуальную подсказку кода, возможность тут же выполнить созданный скрипт, а также многое другое.

Для Python можно использовать различные среды разработки, но одной из самых популярных из них является среда PyCharm, созданная компанией JetBrains. Эта среда динамично развивается, постоянно обновляется и доступна для наиболее распространенных операционных систем - Windows, MacOS, Linux.

При первом запуске открывается начальное окно:

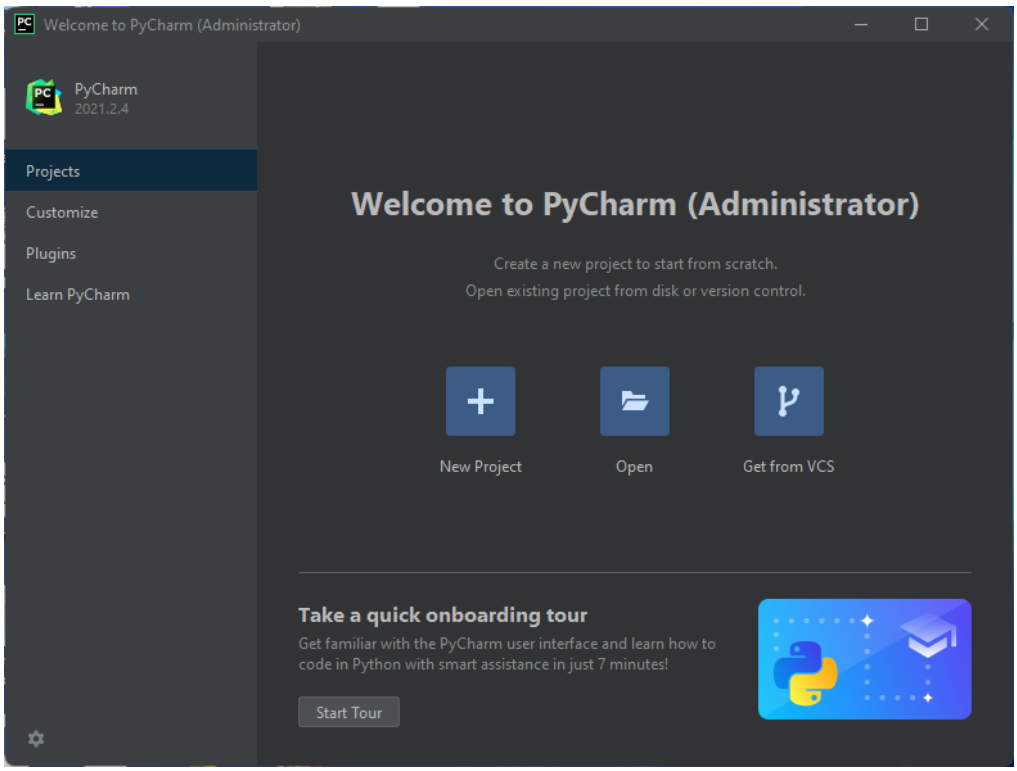

Рис. 4. Создание проекта в IDE PyCharm

Создадим проект и для этого выберем пункт New Project.

Далее откроется окно для настройки проекта. В поле Location необходимо указать путь к проекту. Название директории и будет названием проекта.

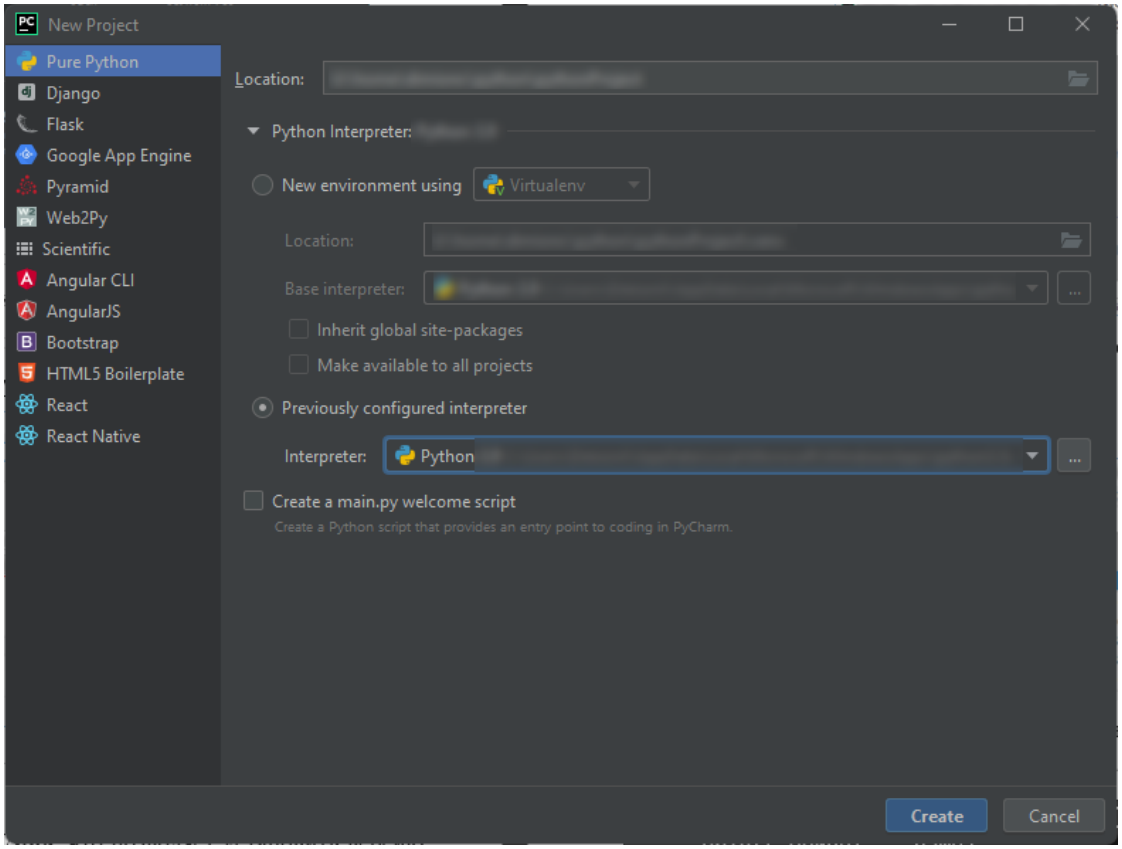

Рис. 5. Создание проекта в IDE PyCharm

Следует отметить, что ***PyCharm*** позволяет разграничить настройки проектов. Так, по умолчанию выбрано поле ***New Environment Using***, что позволяет установить версию интерпретатора для конкретного проекта. Затем все устанавливаемые дополнительные пакеты будут касаться только текущего проекта. Это удобно, если мы создаем несколько проектов, но каждый из которых работает со специфической версией интерпретатора. Но в качестве альтернативы мы также можем выбрать поле ***Existing Interpreter*** и задать путь к файлу интерпретатора глобально для всех проектов.

В реальности для первого простейшего приложения на ***PyCharm*** не имеет значения, как будет установлен интерпретатор. Оставьте выбранный по умолчанию флажок ***New Environment Using*** и под ним в поле ***Base Interpreter*** укажите путь к файлу интерпретатора, установка которого рассматривалась в первой теме.

После установки всех путей нажмите на кнопку ***Create*** для создания проекта.

После этого будет создан пустой проект:

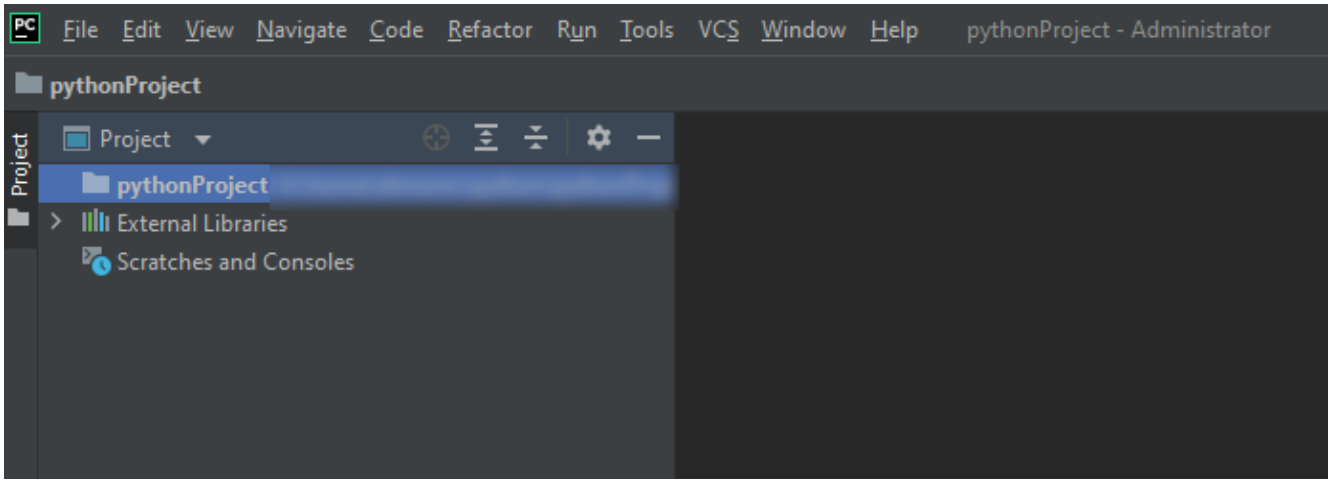

Рис. 6. Пустой проект

Создайте простейшую программу. Для этого нажмите на название проекта правой кнопкой мыши и в появившемся контекстном меню выберем ***New → Python File***.

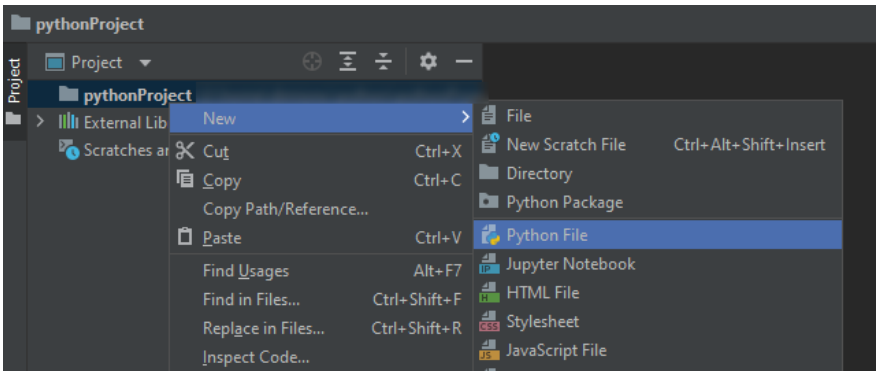

Рис. 7. Создание скрипта

В созданный файл введем следующие строки:

```
name = input("Введите ваше имя: ")
print("Привет,", name)
```

Для запуска скрипта нажмите на него правой кнопкой мыши и в контекстном меню выберите **Run** (либо перейдите в меню **Run** и там нажмите на подпункт **Run...**):

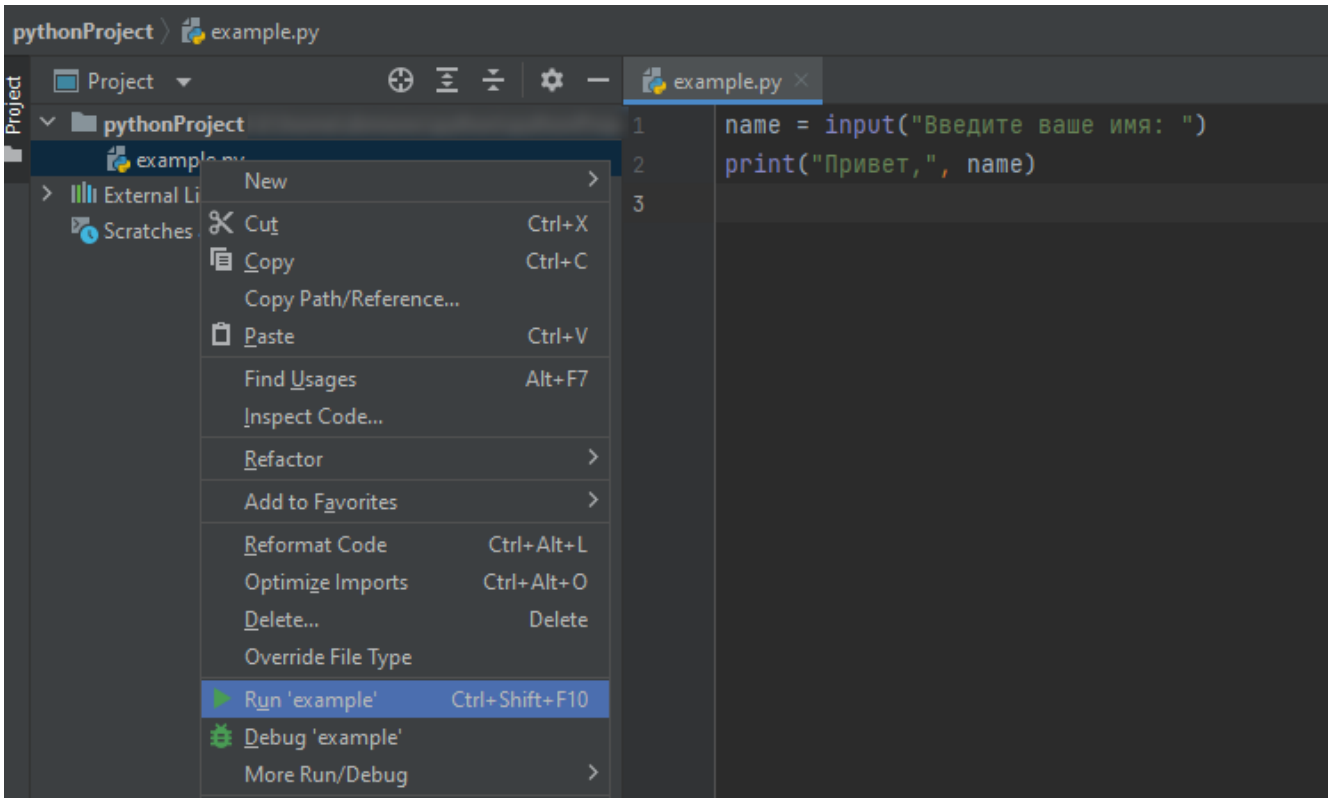

Рис. 8. Запуск скрипта

После этого внизу IDE отобразится окно вывода, где надо будет ввести имя и где после этого будет выведено приветствие:

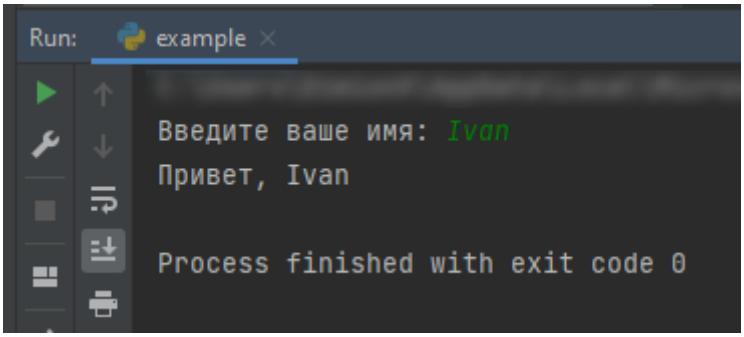

Рис. 9. Результат выполнения скрипта

Для того чтобы установить «Автоматическое форматирование кода при сохранении», нужно зайти в Settings => Tools => Actions on Save и выбрать «Reformat code».

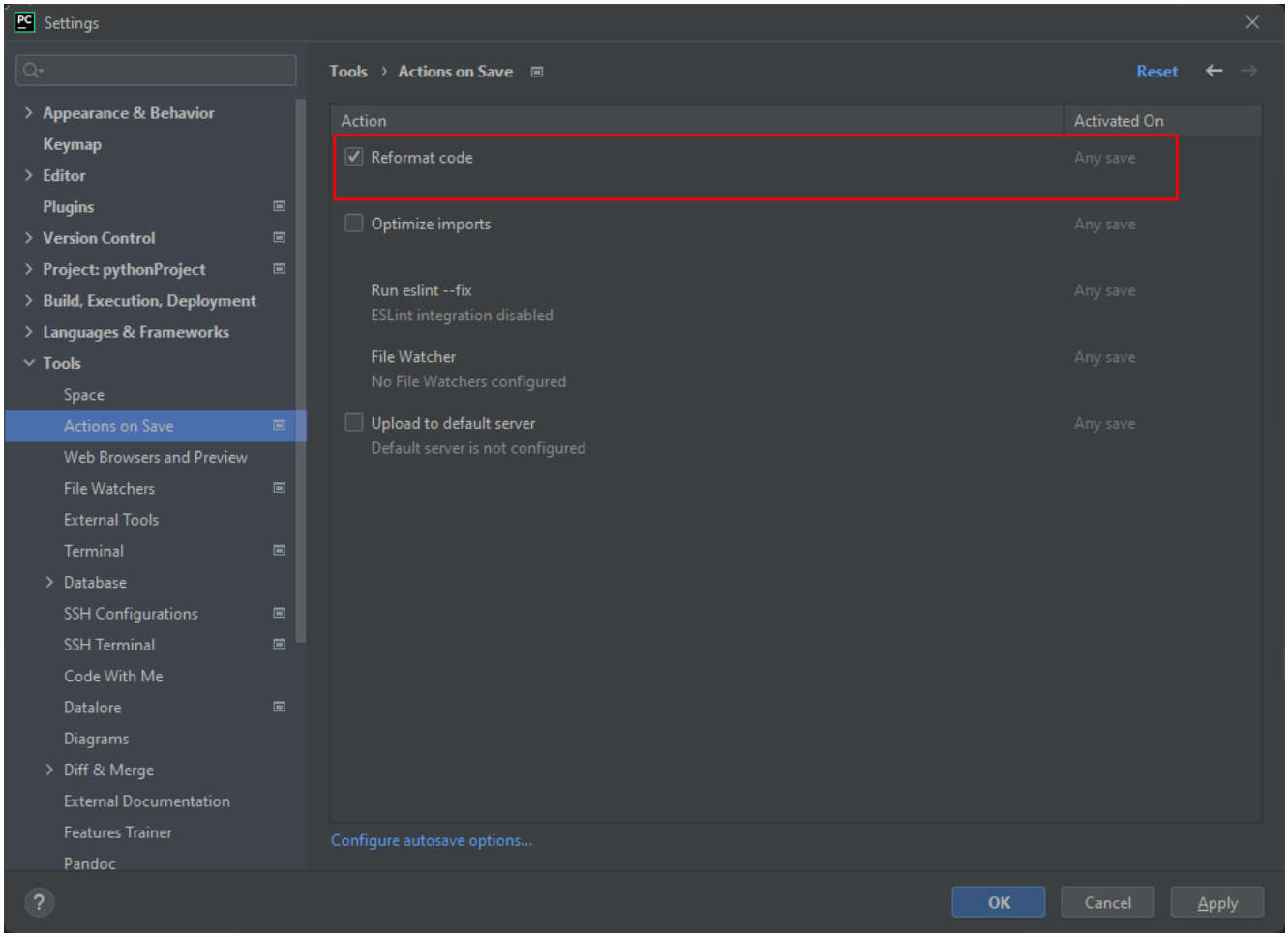

Рис. 10. Автоматическое форматирование кода при сохранении

Программа на языке Python

Программа на языке Python состоит из набора инструкций. Каждая ***инструкция*** помещается на новую строку. Например:

```
print(2 + 3)
print("Hello")
```

Инструкция – элемент языка, определяющий действие, которое требуется выполнить.

Большую роль в Python играют отступы. Неправильно поставленный отступ фактически является ошибкой. Например, в следующем случае мы получим ошибку, хотя код будет практически аналогичен приведенному выше:

```
print(2 + 3)
    print("Hello")
```

Python – регистрозависимый язык, поэтому выражения ***print*** и ***Print***, или ***PRINT*** представляют разные выражения. И если вместо метода ***print*** для вывода на консоль мы попробуем использовать метод Print:

```
>>> Print("hello world")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Print' is not defined
```

Комментарии

Комментарии – вспомогательные строки, не обрабатываемые программой, обозначаются знаком **#** перед началом строки и действуют до конца строки. Зачастую бывает полезно записать комментарий, чтобы оставить напоминание о том, что выполняется в программе в данный момент.

Блочные комментарии ставятся в начале строки:

```
# Вывод сообщения на консоль
print("Hello World")
```

Строчные комментарии располагаются на той же строке, что и инструкции языка:

```
print("Hello World")  # Вывод сообщения на консоль
```

Многострочный комментарий:

```
"""
Многострочный
комментарий
"""
print("Hello World")
```

В ***PyCharm*** можно выделить блок, который нужно закомментировать, и нажать `Ctrl` + `/`. Чтобы раскомментировать блок, нужно опять же нажать `Ctrl` + `/`.

Работа с переменными

Переменная хранит определенные данные. Название переменной в Python должно начинаться с алфавитного символа или со знака подчеркивания и может содержать алфавитно-цифровые символы и знак подчеркивания. И кроме того, название переменной не должно совпадать с названием ключевых слов языка Python.

Например, создадим переменную:

```
name = "Tom"
```

Здесь определена переменная `name`, которая хранит строку `"Tom"`.

При написании кода иногда приходится использовать две, три и даже больше переменных. Каждая из них должна иметь свой уникальный идентификатор. Этот идентификатор может быть практически любым, однако стоит придерживаться некоторых правил, которые позволят легче перечитывать свой собственный код. Представьте, что вы написали некую программу, оставили ее на какое-то время и спустя месяц решили внести дополнения. Если вы называли ваши переменные как попало, тогда вам понадобится потрудиться, чтобы вспомнить что происходит внутри кода. Гораздо удобней, когда в имя переменной закладывается ее смысл.

В стандартном пакете Python 3 существуют ряд встроенных выражений, которым отведен определенный функционал. Ознакомьтесь с ними, чтобы в будущем избежать совпадений в названии своих переменных – https://docs.python.org/3/library/functions.html.

Переменная хранит данные одного из типов данных. В Python существует множество различных типов данных, которые подразделяются на категории: числа, последовательности, словари, наборы:

- ***boolean*** – логическое значение `True` или `False`;
- ***int*** – представляет целое число, для хранения которого используется 4 байта в памяти компьютера;
- ***float*** – представляет число с плавающей точкой, для хранения которого используется 8 байт, например, `1.2` или `34.76`;
- ***complex*** – комплексные числа – `2+3j`;
- ***str*** – строки, например `"hello"`. В Python 3.x строки представляют собой набор символов в кодировке Unicode;
- ***bytes*** – последовательность чисел в диапазоне от 0 до 255;
- ***byte array*** – массив байтов, аналогичен bytes с тем отличием, что может изменяться;
- ***list*** – список;
- ***tuple*** – кортеж;
- ***set*** – неупорядоченная коллекция уникальных объектов;
- ***frozen set*** – то же самое, что и ***set***, только не может изменяться (*immutable*);
- ***dict*** – словарь, где каждый элемент имеет ключ и значение.

Python является языком с динамической типизацией. Он определяет тип данных переменной исходя из значения, которое ей присвоено. Так, при присвоении строки в двойных или одинарных кавычках переменная имеет тип ***str***. При присвоении целого числа ***Python*** автоматически определяет тип переменной как ***int***. Чтобы определить переменную как объект ***float***, ей присваивается дробное число, в котором разделителем целой и дробной части является точка. Число с плавающей точкой можно определять в экспоненциальной записи:

```
x = 3.9e3
print(x)  # 3900.0

x = 3.9e-3
print(x)  # 0.0039
```

Число *float* может иметь только 18 значимых символов. Так, в данном случае используются только два символа - 3.9. И если число слишком велико или слишком мало, то мы можем записывать число в подобной нотации, используя экспоненту. Число после экспоненты указывает степень числа 10, на которое надо умножить основное число - 3.9.

При этом в процессе работы программы мы можем изменить тип переменной, присвоив ей значение другого типа:

```
user_id = "qwerty"  # тип str
print(user_id)

user_id = 234  # тип int
print(user_id)
```

С помощью функции **type()** динамически можно узнать текущий тип переменной:

```
user_id = "qwerty"
print(type(user_id))  # <class 'str'>

user_id = 234
print(type(user_id))  # <class 'int'>
```

Полезная информация

Python: https://www.python.org/downloads/

Python IDE:

- «PyCharm» https://www.jetbrains.com/pycharm/
  - Лицензия для студентов – https://www.jetbrains.com/student/
- «Spyder» https://github.com/spyder-ide/spyder
- Visual Studio (+ Python Tools for Visual Studio) – https://visualstudio.microsoft.com/ru/vs/
  - Для студентов – https://azure.microsoft.com/ru-ru/education/

Документация: https://www.python.org/doc/

Конференции:

- «Pycon Russia» https://pycon.ru/
- Moscow Python Meetup http://www.moscowpython.ru/
- Python Events https://www.python.org/events/

Вакансии (примеры):

- Разработчик-исследователь беспилотных автомобилей
- Python-разработчик аналитической системы
- Разработчик-аналитик (большие данные)
- Разработчик инфраструктуры высоконагруженных систем
- Разработчик в службу сетевого балансировщика
- Разработчик синтеза речи в Алису
- Web-разработчик