


Теперь доступна версия 1.82 (/updates)! Читайте о новых функциях и исправлениях от августа. ✕

Topics

In this article

 (<https://vscode.dev/github/microsoft/vscode-docs/blob/main/docs/python/debugging.md>)

## Отладка Python в VS Code

Расширение Python поддерживает отладку нескольких типов приложений Python. Краткое пошаговое руководство по базовой отладке см. в руководстве - Настройка и запуск отладчика (/docs/python/python-tutorial#\_configure-and-run-the-debugger). Также смотрите руководство по Flask (/docs/python/tutorial-flask). Оба руководства демонстрируют основные навыки, такие как установка точек останова и пошаговое выполнение кода.

Для получения общих функций отладки, таких как проверка переменных, установка точек останова и другие действия, не зависящие от языка, просмотрите VS Code debugging (/docs/editor/debugging).

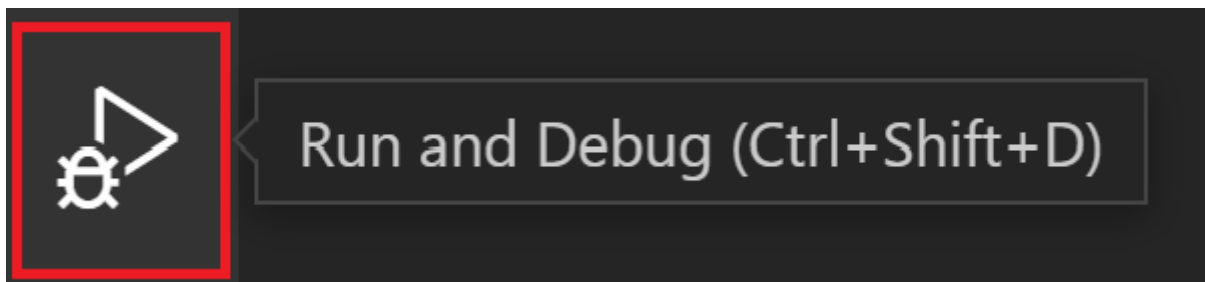
В этой статье в основном рассматриваются *конфигурации* отладки, специфичные для Python, включая необходимые шаги для определенных типов приложений и удаленной отладки.

### Инициализация конфигураций

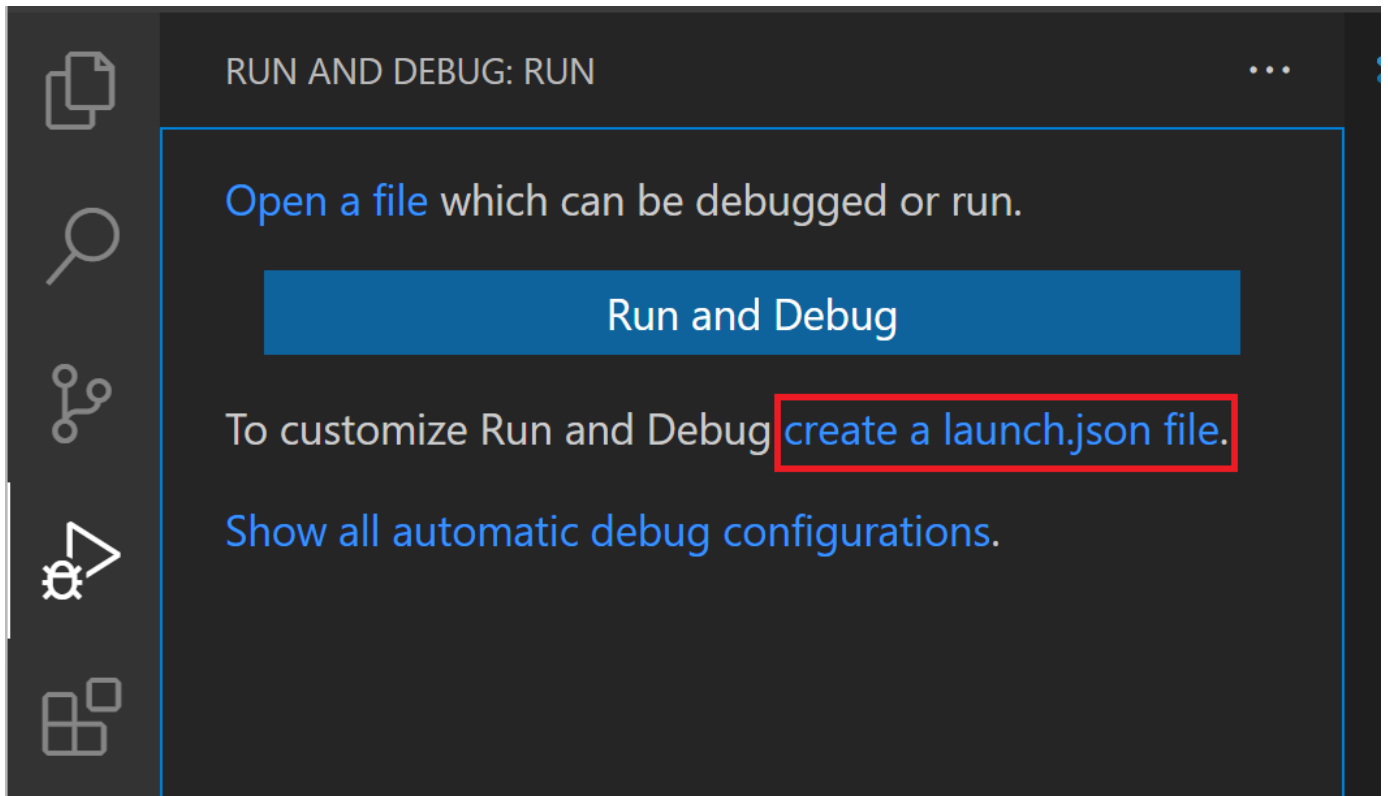
Конфигурация определяет поведение VS Code во время сеанса отладки. Конфигурации определяются в `launch.json` файле, который хранится в `.vscode` папке в вашей рабочей области.

**Примечание:** чтобы изменить конфигурацию отладки, ваш код должен храниться в папке.

Чтобы инициализировать конфигурации отладки, сначала выберите представление **Выполнить** на боковой панели:

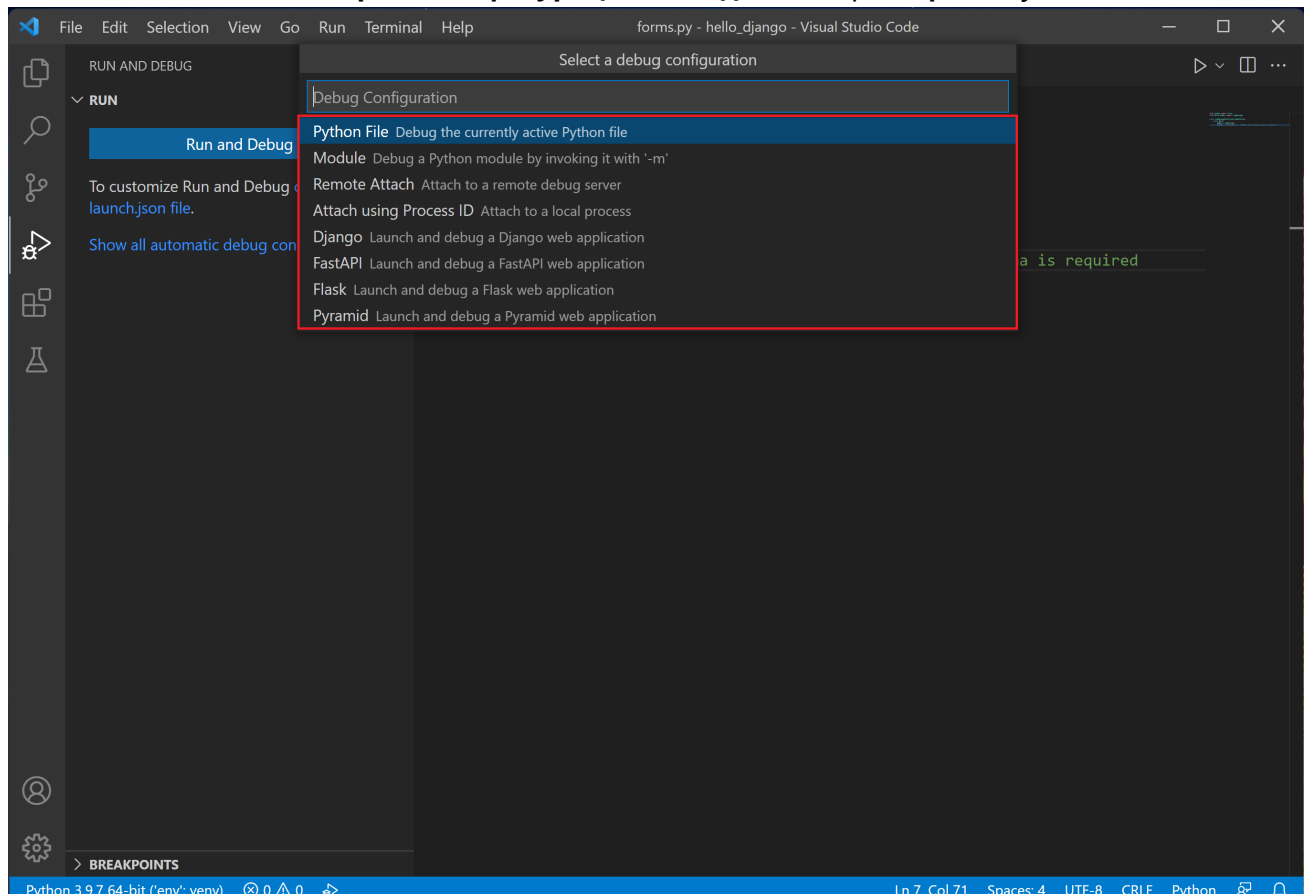


Если у вас еще не определены какие-либо конфигурации, вы увидите кнопку для **запуска и отладки** и ссылку для создания файла конфигурации (`launch.json`):



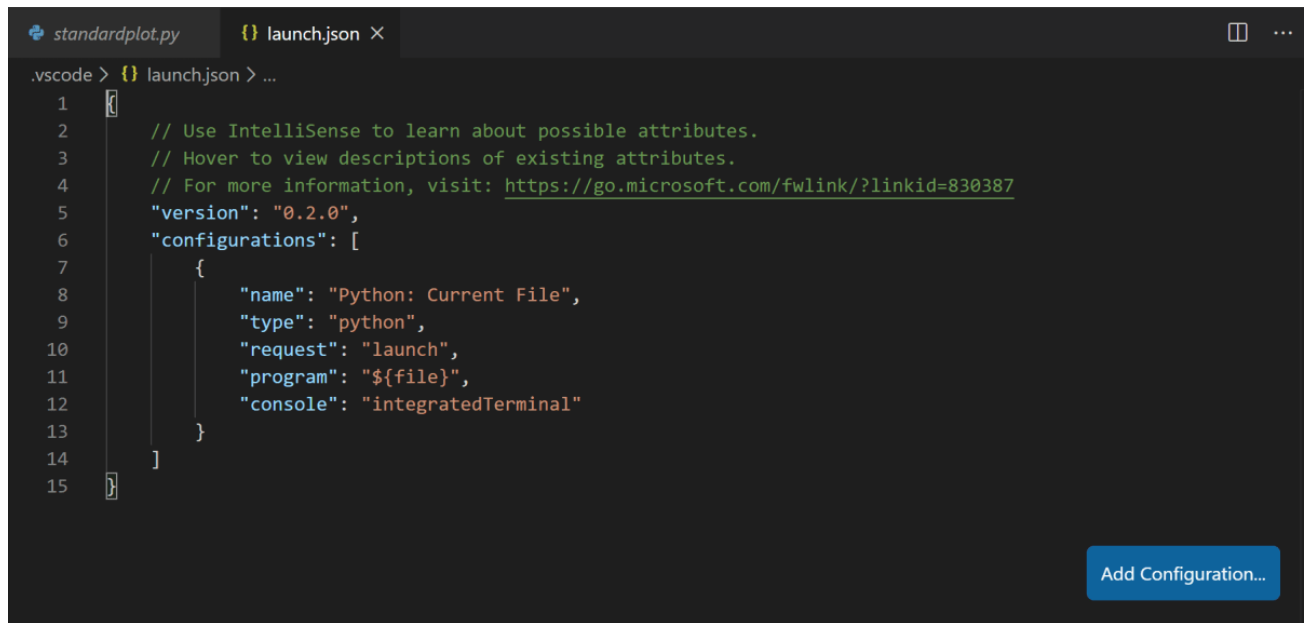
Чтобы сгенерировать `launch.json` файл с конфигурациями Python, выполните следующие действия:

1. Выберите ссылку **создать файл launch.json** (показана на рисунке выше) или используйте команду меню **Выполнить > Открыть конфигурации**.
2. Из командной палитры откроется меню конфигурации, позволяющее вам выбрать тип конфигурации отладки, который вы хотите для открытого файла. На данный момент в появившемся меню **Выберите конфигурацию отладки** выберите **файл Python**.



**Примечание:** Запуск сеанса отладки через панель отладки, F5 или **Выполнить > Начать отладку**, когда конфигурация не существует, также вызовет меню конфигурации отладки, но не создаст файл `launch.json`.

3. Затем расширение Python создает и открывает `launch.json` файл, содержащий предварительно определенную конфигурацию, основанную на том, что вы ранее выбрали, в данном случае **файл Python**. Вы можете изменять конфигурации (например, добавлять аргументы), а также добавлять пользовательские конфигурации.



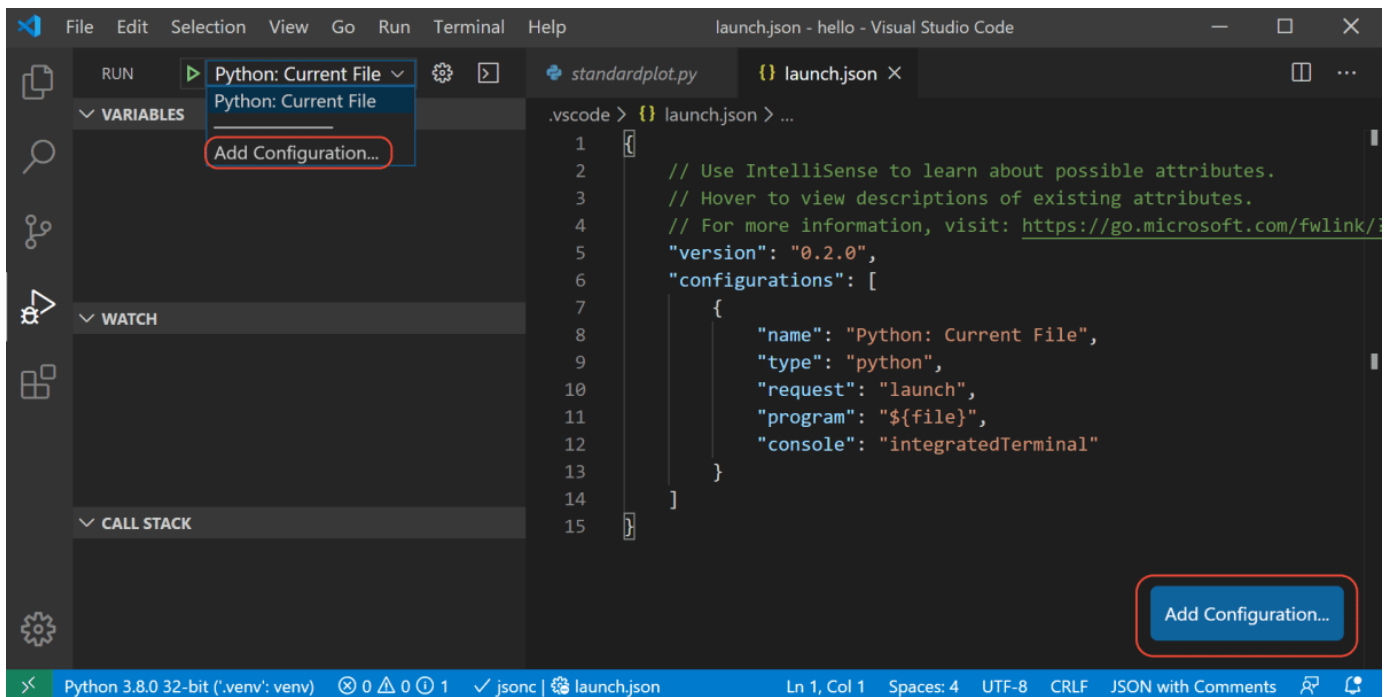
```
.vscode > {} launch.json > ...
1
2 // Use IntelliSense to learn about possible attributes.
3 // Hover to view descriptions of existing attributes.
4 // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5 "version": "0.2.0",
6 "configurations": [
7   {
8     "name": "Python: Current File",
9     "type": "python",
10    "request": "launch",
11    "program": "${file}",
12    "console": "integratedTerminal"
13  }
14 ]
15
```

Add Configuration...

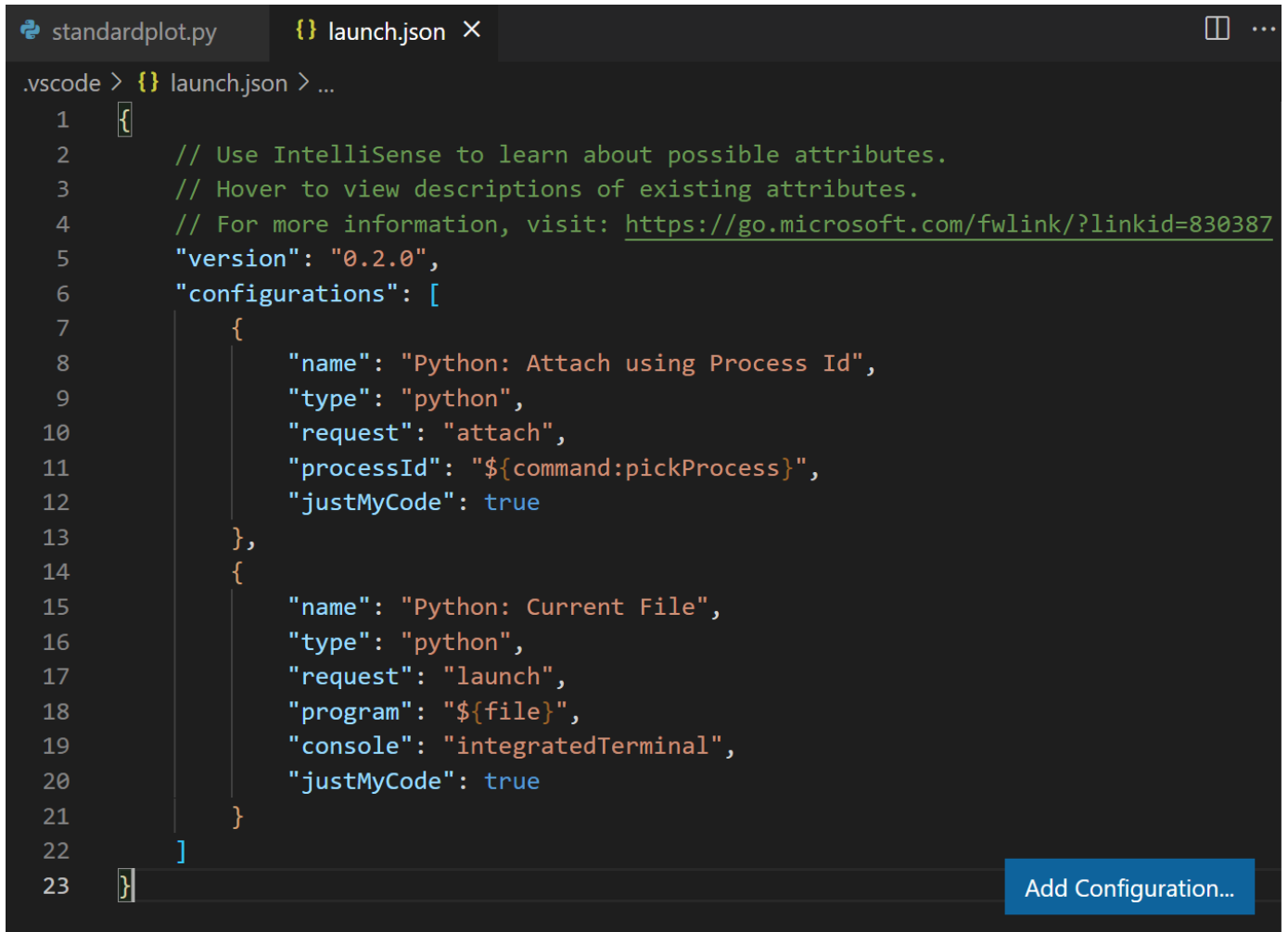
Подробности свойств конфигурации рассматриваются далее в этой статье в разделе Стандартная конфигурация и параметры. Другие конфигурации также описаны в этой статье в разделе Отладка конкретных типов приложений.

## Дополнительные конфигурации

По умолчанию VS Code отображает только наиболее распространенные конфигурации, предоставляемые расширением Python. Вы можете выбрать другие конфигурации для включения `launch.json` с помощью команды **Добавить конфигурацию**, показанной в списке и `launch.json` редакторе. При использовании команды VS Code выдает список всех доступных конфигураций (обязательно выберите опцию **Python**):

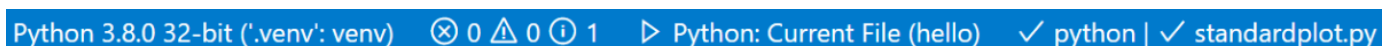


Выбор параметра **Прикрепить с использованием идентификатора процесса** приводит к следующему результату:



Подробнее обо всех этих конфигурациях см. в разделе [Отладка конкретных типов приложений](#).

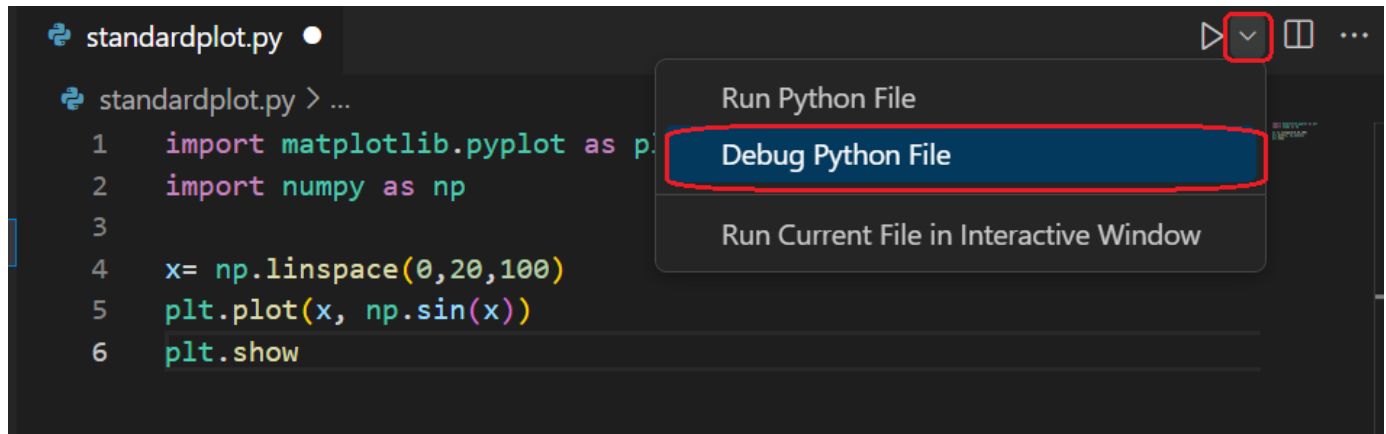
Во время отладки в строке состояния отображаются текущая конфигурация и текущий интерпретатор отладки. При выборе конфигурации отображается список, из которого можно выбрать другую конфигурацию:



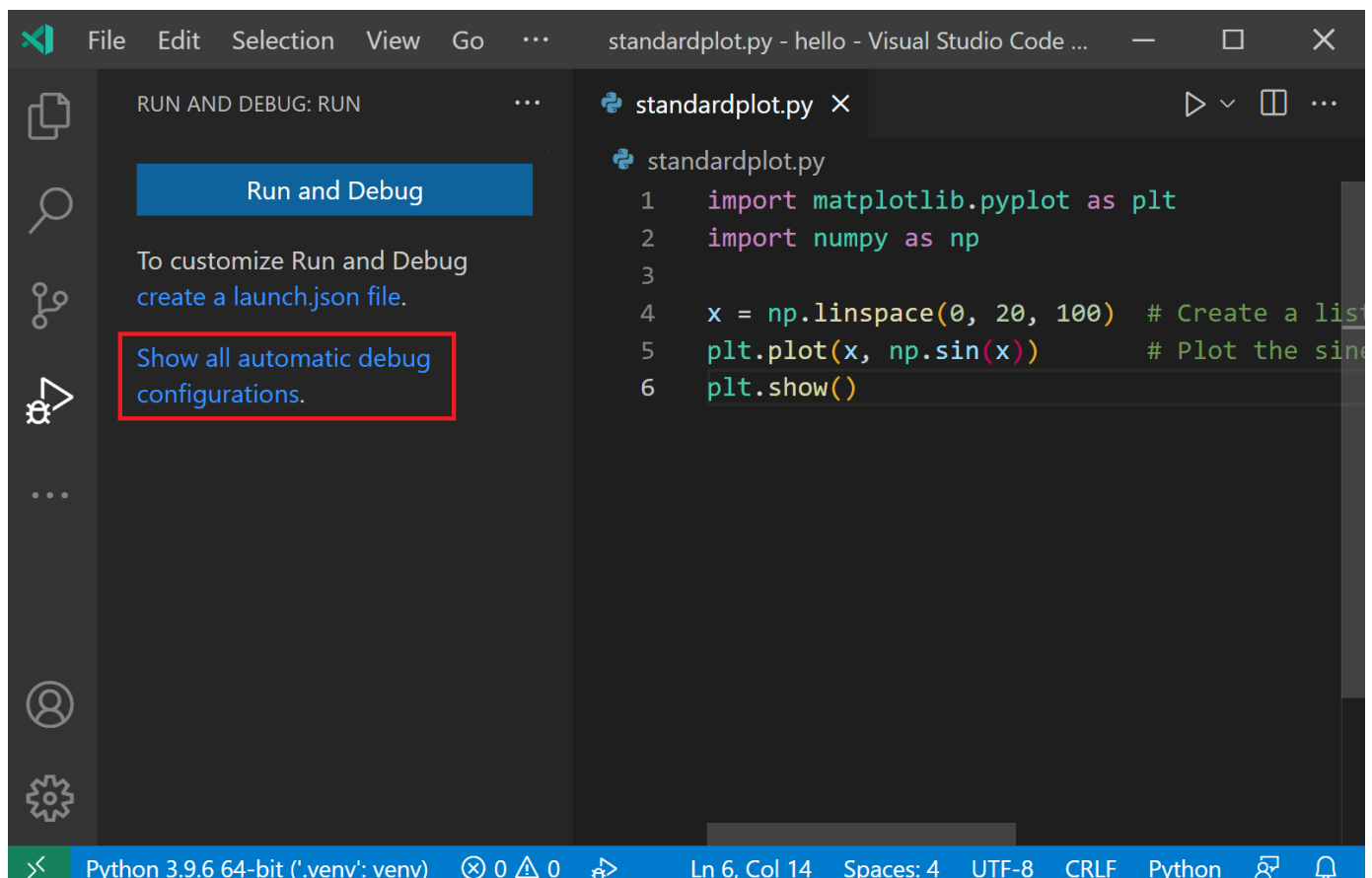
По умолчанию отладчик использует тот же интерпретатор, который выбран для вашей рабочей области, точно так же, как и другие функции расширения Python для VS Code. Чтобы использовать другой интерпретатор конкретно для отладки, задайте значение `python` в `launch.json` для соответствующей конфигурации отладчика. В качестве альтернативы, выберите именованный интерпретатор в строке состояния, чтобы выбрать другой.

## Базовая отладка

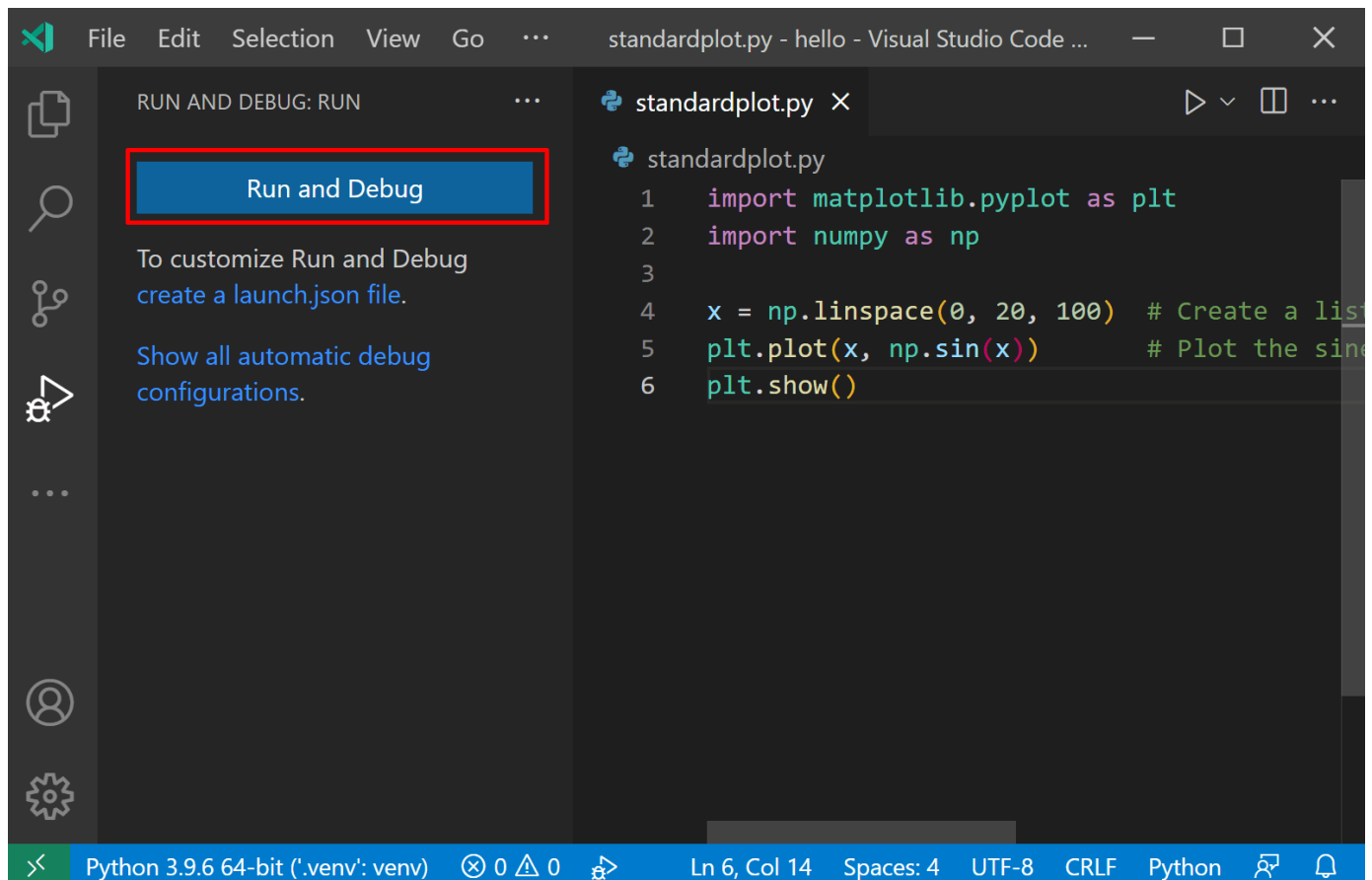
Если вас интересует только отладка скрипта на Python, самый простой способ - нажать стрелку вниз рядом с кнопкой выполнить в редакторе и выбрать **Отладить файл Python в терминале**.



Если вы хотите отладить веб-приложение с помощью Flask, Django или FastAPI, расширение Python предоставляет динамически создаваемые конфигурации отладки на основе структуры вашего проекта с помощью опции **Показать все конфигурации автоматической отладки** в представлении "Запуск и отладка".




Но если вы хотите отлаживать приложения других типов, вы можете запустить отладчик через представление **"Выполнить"**, нажав на кнопку **"Выполнить и отладить"**.



Если конфигурация не задана, вам будет предоставлен список параметров отладки. Здесь вы можете выбрать подходящий параметр для быстрой отладки вашего кода.

Два распространенных варианта - использовать конфигурацию **файла Python** для запуска текущего открытого файла Python или использовать конфигурацию **Attach using Process ID** для подключения отладчика к процессу, который уже запущен.

Сведения о создании и использовании конфигураций отладки см. в разделах **Инициализация конфигураций** и **Дополнительные конфигурации**. После добавления конфигурации ее можно выбрать из выпадающего списка и запустить с помощью кнопки **Начать отладку**.

 Начать отладку

## Отладка командной строки

Отладчик также можно запустить из командной строки. Синтаксис командной строки отладчика следующий:

```
python -m debugpy
  --listen | --connect
  [<host>:]<port>
  [--wait-for-client]
  [--configure-<name> <value>]...
  [--log-to <path>] [--log-to-stderr]
  <filename> | -m <module> | -c <code> | --pid <pid>
  [<arg>]...
```

В качестве примера из командной строки вы могли бы запустить отладчик, используя указанный порт (5678), и скрипт, используя следующий синтаксис. В этом примере предполагается, что скрипт выполняется долго, и флаг `--wait-for-client` в нем опущен, что означает, что скрипт не будет ждать подключения клиента.

```
python -m debugpy --listen 5678 ./myscript.py
```

Затем вы должны использовать следующую конфигурацию для подключения из расширения VS Code Python.

```
{
  "name": "Python: Attach",
  "type": "python",
  "request": "attach",
  "connect": {
    "host": "localhost",
    "port": 5678
  }
}
```

**Примечание:** Указание хоста необязательно для **прослушивания**, по умолчанию используется 127.0.0.1.

Если вы хотите отладить удаленный код или код, запущенный в контейнере docker, на удаленной машине или контейнере, вам нужно будет изменить предыдущую команду CLI, чтобы указать хост.

```
python -m debugpy --listen 0.0.0.0:5678 ./myscript.py
```

Тогда связанный файл конфигурации будет выглядеть следующим образом.

```
{
  "name": "Attach",
  "type": "python",
  "request": "attach",
  "connect": {
    "host": "remote-machine-name", // replace this with remote machine name
    "port": 5678
  }
}
```

**Примечание:** Имейте в виду, что при указании значения хоста, отличного от 127.0.0.1 или localhost, вы открываете порт, разрешающий доступ с любого компьютера, что создает риски для безопасности. При выполнении удаленной отладки следует убедиться, что вы принимаете соответствующие меры предосторожности, такие как использование SSH-туннелей.

Отметить	Опции	Описание
<b>--прослушивание</b> или <b>--подключение</b>	[<host>:] <port>	<b>Требуется.</b> Указывает адрес хоста и порт для сервера адаптера отладки для ожидания входящих подключений (--listen) или для соединения с клиентом, который ожидает входящее соединение (--connect). Это тот же адрес, который используется в конфигурации отладки VS Code. По умолчанию адресом хоста является localhost (127.0.0.1) .
<b>--ожидание клиента</b>	Нет	<b>Необязательно.</b> Указывает, что код не должен выполняться до тех пор, пока не будет установлено соединение с сервером отладки. Этот параметр позволяет выполнять отладку с первой строки вашего кода.
<b>--войдите в</b>	<path>	<b>Необязательно.</b> Указывает путь к существующему каталогу для сохранения журналов.
<b>--вход в stderr</b>	Нет	<b>Необязательно.</b> Позволяет debugpy записывать журналы непосредственно в stderr.
<b>--pid</b>	<pid>	<b>Необязательно.</b> Указывает процесс, который уже запущен для внедрения в сервер отладки.
<b>--настроить- &lt;имя&gt;</b>	<value>	<b>Необязательно.</b> Задаёт свойство debug, которое должно быть известно серверу отладки до подключения клиента. Такие свойства можно использовать непосредственно в конфигурации <i>запуска</i> , но они должны быть установлены таким образом для конфигураций <i>прикрепления</i> . Например, если вы не хотите, чтобы сервер отладки автоматически внедрялся в подпроцессы, созданные процессом, к которому вы подключаетесь, используйте <code>--configure-subProcess false</code> .

**Примечание:** [<arg>] может использоваться для передачи аргументов командной строки запускаемому приложению.

## Отладка путем подключения по сетевому соединению

### Отладка локального скрипта

Могут быть случаи, когда вам необходимо отладить скрипт Python, который вызывается локально другим процессом. Например, вы можете отлаживать веб-сервер, который запускает разные скрипты Python для определенных заданий обработки. В таких случаях вам необходимо подключить отладчик VS Code к сценарию после его запуска:

1. Запустите VS Code, откройте папку или рабочую область, содержащую скрипт, и создайте `launch.json` для этой рабочей области, если она еще не существует.
2. В коде скрипта добавьте следующее и сохраните файл:



```
import debugpy

# 5678 is the default attach port in the VS Code debug configurations. Unless
a host and port are specified, host defaults to 127.0.0.1
debugpy.listen(5678)
print("Waiting for debugger attach")
debugpy.wait_for_client()
debugpy.breakpoint()
print('break on this line')
```

3. Откройте терминал с помощью **Terminal: создайте новый терминал**, который активирует выбранную среду скрипта.
4. В терминале установите пакет debugpy с помощью `python -m pip install --upgrade debugpy`.
5. В терминале запустите Python с помощью скрипта, например, `python3 myscript.py`. Вы должны увидеть сообщение "Ожидание подключения отладчика", которое включено в код, и скрипт останавливается при `debugpy.wait_for_client()` вызове.
6. Переключитесь в режим **Запуска и отладки** (Ctrl + Shift + D), выберите соответствующую конфигурацию из выпадающего списка debugger и запустите отладчик.
7. Отладчик должен остановиться при `debugpy.breakpoint()` вызове, после чего вы сможете использовать отладчик в обычном режиме. У вас также есть возможность устанавливать другие точки останова в коде скрипта, используя пользовательский интерфейс вместо использования `debugpy.breakpoint()`.

### Удаленная отладка скрипта с помощью SSH

Удаленная отладка позволяет выполнять пошаговое выполнение программы локально в VS Code во время ее выполнения на удаленном компьютере. Устанавливать VS Code на удаленный компьютер не обязательно. Для дополнительной безопасности при отладке может потребоваться безопасное подключение к удаленному компьютеру, например SSH.

**Примечание:** На компьютерах с Windows может потребоваться установить Windows 10 OpenSSH ([https://learn.microsoft.com/windows-server/administration/openssh/openssh\\_install\\_firstuse](https://learn.microsoft.com/windows-server/administration/openssh/openssh_install_firstuse)), чтобы получить ssh команду.

Следующие шаги описывают общий процесс настройки SSH-туннеля. SSH-туннель позволяет вам работать на вашем локальном компьютере так, как если бы вы работали непосредственно на удаленном компьютере, более безопасным способом, чем если бы порт был открыт для общего доступа.

#### На удаленном компьютере:

1. Включите переадресацию портов, открыв `sshd_config` файл конфигурации (находится в разделе `/etc/ssh/` в Linux и в разделе `%programfiles(x86)%/openssh/etc` в Windows) и добавив или изменив следующий параметр:

```
AllowTcpForwarding yes
```

**Примечание:** значение по умолчанию для `AllowTcpForwarding` равно `yes`, поэтому вам может не понадобиться вносить изменения.

2. Если вам пришлось добавлять или изменять `AllowTcpForwarding`, перезапустите SSH-сервер. В Linux / macOS выполните `sudo service ssh restart`; в Windows выполните `services.msc`, выберите OpenSSH или `sshd` в списке служб и выберите **Перезапустить**.

### На локальном компьютере:

1. Создайте SSH-туннель, запустив `ssh -2 -L sourceport:localhost:destinationport -i identityfile user@remoteaddress`, используя выбранный порт для `destinationport` и соответствующее имя пользователя и IP-адрес удаленного компьютера в `user@remoteaddress`. Например, чтобы использовать порт 5678 на IP-адресе 1.2.3.4, команда будет `ssh -2 -L 5678:localhost:5678 -i identityfile user@1.2.3.4`. Вы можете указать путь к файлу идентификации, используя `-i` флаг.
2. Убедитесь, что вы видите приглашение в сеансе SSH.
3. В рабочей области VS Code создайте конфигурацию для удаленной отладки в своем `launch.json` файле, установив порт, соответствующий порту, используемому в `ssh` команде, а хост - в `localhost`. Вы используете `localhost` здесь, потому что вы настроили SSH-туннель.

```
{
  "name": "Python: Attach",
  "type": "python",
  "request": "attach",
  "port": 5678,
  "host": "localhost",
  "pathMappings": [
    {
      "localRoot": "${workspaceFolder}", // Maps C:\Users\user1\project1
      "remoteRoot": "." // To current working directory ~/project1
    }
  ]
}
```

### Запуск отладки

Теперь, когда для удаленного компьютера настроен SSH-туннель, вы можете начать отладку.

1. На обоих компьютерах: убедитесь, что доступен идентичный исходный код.
2. На обоих компьютерах: установите `debugpy` (<https://pypi.org/project/debugpy/>) с помощью `python -m pip install --upgrade debugpy` в свою среду (хотя использование виртуальной среды не требуется, это рекомендуемая наилучшая практика).
3. Удаленный компьютер: существует два способа указать, как подключиться к удаленному процессу.
  1. В исходном коде добавьте следующие строки, заменив `address` IP-адрес удаленного компьютера и номер порта (IP-адрес 1.2.3.4 показан здесь только для иллюстрации).

```
import debugpy

# Allow other computers to attach to debugpy at this IP address and port.
debugpy.listen(('1.2.3.4', 5678))

# Pause the program until a remote debugger is attached
debugpy.wait_for_client()
```

IP-адрес, используемый в `listen`, должен быть частным IP-адресом удаленного компьютера. Затем вы можете запустить программу в обычном режиме, приостановив ее до подключения отладчика.

2. Запустите удаленный процесс через `debugpy`, например:

```
python3 -m debugpy --listen 1.2.3.4:5678 --wait-for-client -m myproject
```

Это запускает пакет, `myproject` используя `python3` частный IP-адрес удаленного компьютера `1.2.3.4` и прослушивающий порт `5678` (вы также можете запустить удаленный процесс Python, указав путь к файлу вместо использования `-m`, например `./hello.py`).

4. Локальный компьютер: **Только если вы изменили исходный код на удаленном компьютере, как описано выше**, тогда в исходный код добавьте закомментированную копию того же кода, добавленного на удаленном компьютере. Добавление этих строк гарантирует, что исходный код на обоих компьютерах соответствует построчно.

```
#import debugpy

# Allow other computers to attach to debugpy at this IP address and port.
#debugpy.listen(('1.2.3.4', 5678))

# Pause the program until a remote debugger is attached
#debugpy.wait_for_client()
```

5. Локальный компьютер: переключитесь в режим **запуска и отладки** (`Ctrl + Shift + D`) в VS Code, выберите конфигурацию **Python: Прикрепить**

6. Локальный компьютер: установите точку останова в коде, с которого вы хотите начать отладку.

7. Локальный компьютер: запустите отладчик VS Code, используя измененную конфигурацию **Python: Attach** и кнопку Начать отладку. VS Code должен останавливаться на ваших локально установленных точках останова, позволяя вам пошагово просматривать код, проверять переменные и выполнять все другие действия по отладке. Выражения, которые вы вводите в **консоль отладки**, также выполняются на удаленном компьютере.

Текстовый вывод в стандартный вывод, как из `print` инструкций, отображается на обоих компьютерах. Однако другие выходные данные, такие как графические графики из пакета, подобного `matplotlib`, отображаются только на удаленном компьютере.

8. Во время удаленной отладки панель инструментов отладки отображается следующим образом:



На этой панели инструментов кнопка отключения ( Shift + F5 ) останавливает отладчик и позволяет удаленной программе работать до завершения. Кнопка перезапуска ( Ctrl + Shift + F5 ) перезапускает отладчик на локальном компьютере, но **не** перезапускает удаленную программу. Используйте кнопку перезапуска только тогда, когда вы уже перезапустили удаленную программу и вам необходимо повторно подключить отладчик.

## Задайте параметры конфигурации

При первом создании `launch.json` существуют две стандартные конфигурации, которые запускают активный файл в редакторе либо на встроенном терминале (внутри VS Code), либо на внешнем терминале (вне VS Code):

```
{
  "configurations": [
    {
      "name": "Python: Current File (Integrated Terminal)",
      "type": "python",
      "request": "launch",
      "program": "${file}",
      "console": "integratedTerminal"
    },
    {
      "name": "Python: Current File (External Terminal)",
      "type": "python",
      "request": "launch",
      "program": "${file}",
      "console": "externalTerminal"
    }
  ]
}
```

Конкретные параметры описаны в следующих разделах. Вы также можете добавить другие параметры, такие как `args`, которые не включены в стандартные конфигурации.

**Совет:** В проекте часто бывает полезно создать конфигурацию, которая запускает определенный файл запуска. Например, если вы хотите всегда запускать отладчик `startup.py` с аргументами `--port 1593` при запуске отладчика, создайте запись конфигурации следующим образом:

```
{
  "name": "Python: startup.py",
  "type": "python",
  "request": "launch",
  "program": "${workspaceFolder}/startup.py",
  "args": ["--port", "1593"]
},
```

## Имя

Предоставляет имя конфигурации отладки, которая отображается в раскрывающемся списке VS Code.

## Тип

Определяет тип используемого отладчика; оставьте этот параметр равным `python` для кода Python.

## запрос

Определяет режим, в котором следует начинать отладку:

- `launch` : запустите отладчик в файле, указанном в `program`
- `attach` : подключите отладчик к уже запущенному процессу. Пример см. в разделе Удаленная отладка.

## программа

Предоставляет полный путь к модулю ввода программы python (файл запуска). Значение `${file}` , часто используемое в конфигурациях по умолчанию, использует текущий активный файл в редакторе. Указав конкретный файл запуска, вы всегда можете быть уверены в запуске вашей программы с одной и той же точкой входа, независимо от того, какие файлы открыты. Например:

```
"program": "/Users/Me/Projects/PokemonGo-Bot/pokemongo_bot/event_handlers/__init__.py",
```

Вы также можете полагаться на относительный путь из корня workspace. Например, если корнем является `/Users/Me/Projects/PokemonGo-Bot` , то вы можете использовать следующий пример:

```
"program": "${workspaceFolder}/pokemongo_bot/event_handlers/__init__.py",
```

## модуль

Предоставляет возможность указать имя модуля, подлежащего отладке, аналогично `-m` аргументу при запуске в командной строке. Для получения дополнительной информации см. Python.org (<https://docs.python.org/3/using/cmdline.html#cmdoption-m>)

## python

Полный путь, указывающий на интерпретатор Python, который будет использоваться для отладки.

Если этот параметр не указан, по умолчанию используется интерпретатор, выбранный для вашей рабочей области, что эквивалентно использованию значения

`${command:python.interpreterPath}` . Чтобы использовать другой интерпретатор, укажите его путь вместо этого в `python` свойстве конфигурации отладки.

В качестве альтернативы вы можете использовать пользовательскую переменную среды, определенную на каждой платформе, чтобы содержать полный путь к используемому интерпретатору Python, так что никакие другие пути к папкам не требуются.

Если вам нужно передать аргументы интерпретатору Python, вы можете использовать `pythonArgs` свойство.

### `pythonArgs`

Задаёт аргументы для передачи интерпретатору Python с использованием синтаксиса `"pythonArgs": ["<arg 1>", "<arg 2>", ...]`.

### аргументы

Задаёт аргументы для передачи в программу Python. Каждый элемент строки аргумента, разделённый пробелом, должен содержаться в кавычках, например:

```
"args": ["--quiet", "--norepeat", "--port", "1593"],
```

### `stopOnEntry`

При значении `true` отладчик прерывает работу в первой строке отлаживаемой программы. Если параметр опущен (по умолчанию) или установлено значение `false`, отладчик запускает программу до первой точки останова.

### консоль

Определяет способ отображения выходных данных программы до тех пор, пока значения по умолчанию для `redirectOutput` не будут изменены.

Значение	Где вывод отображается
"internalConsole"	VS Code debug console. Если <code>redirectOutput</code> установлено значение <code>False</code> , выходные данные не отображаются.
"integratedTerminal" (по умолчанию)	Встроенный терминал VS Code (/docs/terminal/basics). Если <code>redirectOutput</code> установлено значение <code>True</code> , выходные данные также отображаются в консоли отладки.
"externalTerminal"	<b>Отдельное окно консоли.</b> Если <code>redirectOutput</code> установлено значение <code>True</code> , выходные данные также отображаются в консоли отладки.

### цель

Существует несколько способов настройки кнопки **Выполнить** с помощью `purpose` опции. Установка параметра в `debug-test` определяет, что конфигурация должна использоваться при отладке тестов в VS Code. Однако установка параметра в `debug-in-terminal` значение определяет, что конфигурация должна использоваться только при доступе к кнопке **"Запустить файл Python"** в правом верхнем углу редактора (независимо от того, используются ли параметры **"Запустить файл**

Python" или "Отладить файл Python", **предоставляемые кнопкой**). **Файл Python. Примечание:** Этот `purpose` параметр нельзя использовать для запуска отладчика с помощью `F5` или **Выполнить > Начать отладку**.

## Автозагрузка

Позволяет автоматически перезагружать отладчик при внесении изменений в код после того, как выполнение отладчика достигло точки останова. Чтобы включить эту функцию, установите `{"enable": true}`, как показано в следующем коде.

```
{
  "name": "Python: Current File",
  "type": "python",
  "request": "launch",
  "program": "${file}",
  "console": "integratedTerminal",
  "autoReload": {
    "enable": true
  }
}
```

**\* Примечание:** Когда отладчик выполняет перезагрузку, код, который выполняется при импорте, может быть выполнен снова. Чтобы избежать этой ситуации, старайтесь использовать в своем модуле только импорт, константы и определения, помещая весь код в функции. В качестве альтернативы вы также можете использовать `if __name__=="__main__"` проверки.

## Подпроцесс

Указывает, следует ли включать отладку подпроцессов. По умолчанию используется значение `false`, для включения установлено значение `true`. Дополнительные сведения см. в разделе [многоцелевая отладка \(/docs/editor/debugging#\\_multitarget-debugging\)](/docs/editor/debugging#_multitarget-debugging).

### cwd

Указывает текущий рабочий каталог для отладчика, который является базовой папкой для любых относительных путей, используемых в коде. Если параметр опущен, по умолчанию используется `${workspaceFolder}` (папка, открытая в VS Code).

В качестве примера, скажем, `${workspaceFolder}` содержит `py_code` папку, содержащую `app.py`, и `data` папку, содержащую `salaries.csv`. Если вы запускаете отладчик на `py_code/app.py`, то относительные пути к файлу данных меняются в зависимости от значения `cwd`:

cwd	Относительный путь к файлу данных
Опущено или <code>\${workspaceFolder}</code>	<code>data/salaries.csv</code>
<code>\${workspaceFolder}/py_code</code>	<code>../data/salaries.csv</code>
<code>\${workspaceFolder}/data</code>	<code>salaries.csv</code>

## Перенаправление вывода

Если установлено значение `true` (по умолчанию для `internalConsole`), отладчик выводит все выходные данные программы в окно вывода отладки VS Code. Если установлено значение `false` (по умолчанию для `integratedTerminal` и `externalTerminal`), выходные данные программы не отображаются в окне вывода отладчика.

Этот параметр обычно отключен при использовании `"console": "integratedTerminal"` or `"console": "externalTerminal"`, поскольку нет необходимости дублировать выходные данные в консоли отладки.

## JustMyCode

Если этот параметр опущен или установлено значение `true` (по умолчанию), отладка ограничивается только пользовательским кодом. Установите значение `false`, чтобы также включить отладку стандартных библиотечных функций.

## django

Если установлено значение `true`, активируются функции отладки, специфичные для Django web Framework.

## sudo

Когда установлено значение `true` и используется `"console": "externalTerminal"`, позволяет отлаживать приложения, требующие прав доступа. Для записи пароля необходимо использовать внешнюю консоль.

## пирамида

При значении `true` гарантирует, что приложение Pyramid будет запущено с помощью необходимой `pserve` команды (<https://docs.pylonsproject.org/projects/pyramid/en/latest/narr/startup.html?highlight=pserve>).

## env

Устанавливает необязательные переменные среды для процесса debugger помимо системных переменных среды, которые отладчик всегда наследует. Значения для этих переменных должны вводиться в виде строк.

## envFile

Optional path to a file that contains environment variable definitions. See [Configuring Python environments - environment variable definitions file \(/docs/python/environments#\\_environment-variable-definitions-file\)](#).



## gevent

Если установлено значение `true`, включается отладка исправленного кода `gevent monkey` (<https://www.gevent.org/intro.html>).

## jinja

Если установлено значение `true`, активируются функции отладки, специфичные для платформы шаблонов Jinja (<https://jinja.palletsprojects.com>).

# Точки останова и входа в систему

Расширение Python поддерживает точки останова ([/docs/editor/debugging#\\_breakpoints](/docs/editor/debugging#_breakpoints)) и точки входа ([/docs/editor/debugging#\\_logpoints](/docs/editor/debugging#_logpoints)) в систему для отладки кода. Краткое пошаговое руководство по базовой отладке и использованию точек останова см. в руководстве - Настройка и запуск отладчика ([/docs/python/python-tutorial#\\_configure-and-run-the-debugger](/docs/python/python-tutorial#_configure-and-run-the-debugger)).

## Условные точки останова

Точки останова также могут запускаться на основе выражений, количества попаданий или комбинации того и другого. Расширение Python поддерживает количество совпадений, которые являются целыми числами, в дополнение к целым числам, которым предшествуют операторы `==`, `>`, `>=`, `<`, `<=`, и `%`. Например, вы могли бы установить точку останова для срабатывания после пяти вхождений, установив количество попаданий, равное `>5`. Для получения дополнительной информации см. условные точки останова ([/docs/editor/debugging#\\_conditional-breakpoints](/docs/editor/debugging#_conditional-breakpoints)) в основной статье об отладке VS Code.

## Вызов точки останова в коде

В вашем коде Python вы можете вызвать `debugpy.breakpoint()` в любой момент, когда вы хотите приостановить работу отладчика во время сеанса отладки.

## Проверка точки останова

Расширение Python автоматически определяет точки останова, которые установлены в неисполняемых строках, таких как `pass` инструкции или середина многострочной инструкции. В таких случаях запуск отладчика перемещает точку останова на ближайшую допустимую строку, чтобы гарантировать, что выполнение кода остановится в этой точке.

# Отладка конкретных типов приложений

Раскрывающийся список конфигурация предоставляет различные варианты для общих типов приложений:

Конфигурация	Описание
Прикрепить	Смотрите Удаленную отладку в предыдущем разделе.
Django	Указывает "program": "\${workspaceFolder}/manage.py" , "args": ["runserver"] . Также добавляет "django": true для включения отладки HTML-шаблонов Django.
Flask	Смотрите отладку Flask ниже.
Gevent	Добавляет "gevent": true к стандартной конфигурации интегрированного терминала.
Пирамида	Удаляет program , добавляет "args": ["\${workspaceFolder}/development.ini"] , добавляет "jinja": true для включения отладки шаблонов и добавляет "pyramid": true для обеспечения запуска программы с помощью необходимой pserve команды ( <a href="https://docs.pylonsproject.org/projects/pyramid/en/latest/narr/startup.html?highlight=pserve">https://docs.pylonsproject.org/projects/pyramid/en/latest/narr/startup.html?highlight=pserve</a> ).

Для удаленной отладки и Google App Engine также необходимы определенные шаги. Подробные сведения об отладочных тестах см. в разделе Тестирование (/docs/python/testing).

Для отладки приложения, требующего прав администратора, используйте "console": "externalTerminal" и "sudo": "True" .

## Отладка Flask

```
{
  "name": "Python: Flask",
  "type": "python",
  "request": "launch",
  "module": "flask",
  "env": {
    "FLASK_APP": "app.py"
  },
  "args": [
    "run",
    "--no-debugger"
  ],
  "jinja": true
},
```

Как вы можете видеть, в этой конфигурации указаны "env": {"FLASK\_APP": "app.py"} и "args": ["run", "--no-debugger"] . "module": "flask" Свойство используется вместо program . (Вы можете увидеть "FLASK\_APP": "\${workspaceFolder}/app.py" в env свойстве, и в этом случае измените конфигурацию, чтобы ссылаться только на имя файла. В противном случае вы можете увидеть ошибки "Не удалось импортировать модуль C", где C - это буква диска.)

"jinja": true Параметр также включает отладку для механизма создания шаблонов Jinja, используемого Flask по умолчанию.

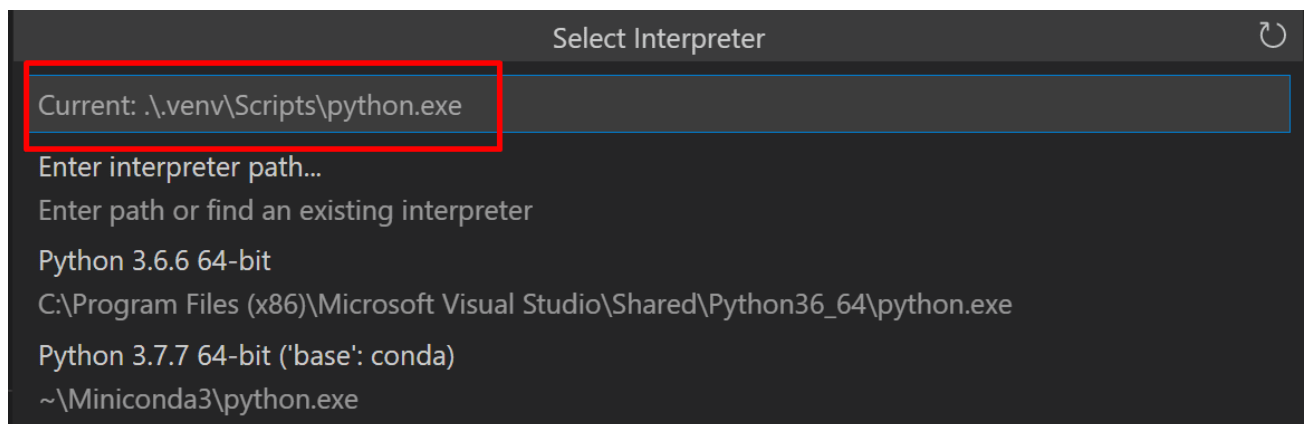
Если вы хотите запустить сервер разработки Flask в режиме разработки, используйте следующую конфигурацию:

```
{
  "name": "Python: Flask (development mode)",
  "type": "python",
  "request": "launch",
  "module": "flask",
  "env": {
    "FLASK_APP": "app.py",
    "FLASK_ENV": "development"
  },
  "args": [
    "run"
  ],
  "jinja": true
},
```

## Устранение неполадок

Существует множество причин, по которым отладчик может не работать. Иногда консоль отладки показывает конкретные причины, но основные причины заключаются в следующем:

- Указан неверный путь к исполняемому файлу python: проверьте путь к выбранному вами интерпретатору, выполнив команду **Python: Select Interpreter** и просмотрев текущее значение:



- В окне просмотра недопустимые выражения: удалите все выражения из окна просмотра и перезапустите отладчик.
- Если вы работаете с многопоточным приложением, которое использует собственные API потоков (например, функцию `Win32 CreateThread`, а не API потоков Python), в настоящее время необходимо включить следующий исходный код в начало любого файла, который вы хотите отладить:

```
import debugpy
debugpy.debug_this_thread()
```

- Если вы работаете с системой Linux, вы можете получить сообщение об ошибке "время ожидания истекло" при попытке применить отладчик к любому запущенному процессу. Чтобы предотвратить это, вы можете временно выполнить следующую команду:

```
echo 0 | sudo tee /proc/sys/kernel/yama/ptrace_scope
```

## Следующие шаги

- Среды Python (/docs/python/environments) - Определяют, какой интерпретатор Python используется для редактирования и отладки.
- Тестирование (/docs/python/testing) - настройка тестовых сред и обнаружение, запуск и отладка тестов.
- Справочник по настройкам (/docs/python/settings-reference) - Изучите полный набор настроек, связанных с Python, в VS Code.
- Общая отладка (/docs/editor/debugging) - узнайте о возможностях отладки VS Code.

### Была ли эта документация полезной?

☐ Да ☐ Нет

1/19/2023

 [Subscribe\(/feed.xml\)](#)  [Ask questions\(https://stackoverflow.com/questions/tagged/vscode\)](https://stackoverflow.com/questions/tagged/vscode)

 [Follow @code\(https://go.microsoft.com/fwlink/?LinkID=533687\)](https://go.microsoft.com/fwlink/?LinkID=533687)

 [Request features\(https://go.microsoft.com/fwlink/?LinkID=533482\)](https://go.microsoft.com/fwlink/?LinkID=533482)

 [Report issues\(https://www.github.com/Microsoft/vscode/issues\)](https://www.github.com/Microsoft/vscode/issues)

 [Watch videos\(https://www.youtube.com/channel/UCs5Y5\\_7XK8HLDX0SLNwkd3w\)](https://www.youtube.com/channel/UCs5Y5_7XK8HLDX0SLNwkd3w)

Привет из Сиэтла. [Следуйте @code \(https://go.microsoft.com/fwlink/?LinkID=533687\)](https://go.microsoft.com/fwlink/?LinkID=533687)

Star 151,012

Поддержка (<https://support.serviceshub.microsoft.com/supportforbusiness/create?sapId=d66407ed-3967-b000-4cfb-2c318cad363d>)

Конфиденциальность (<https://go.microsoft.com/fwlink/?LinkId=521839>)

Условия использования (<https://www.microsoft.com/legal/terms-of-use>) [Лицензия \(/License\)](#)

 Microsoft (<https://www.microsoft.com>)

© 2023 Microsoft