

Первое практическое занятие v.2025.1

Знакомство с курсом и подготовка к работе

Знакомство с курсом

Привет! Мы начинаем увлекательное приключение длиною в год, в нем мы научимся создавать и программировать собственные электронные устройства, поймем как устроены и работают уже существующие, но не стоит переживать и бояться!

ВСЕ ПОЛУЧИТСЯ!



Как достойно пройти весь этот путь? Как использовать эту возможность по максимуму? Что для этого потребуется?

Все просто:

- Нужно не пропускать занятия;
- Спрашивать, если что-то не понятно;
- Быть внимательным и стараться;
- **И главное: ЗАРАНЕЕ** (это значит до того, как пришел на пару) читать методичку к текущему практическому занятию и разбираться в ее материале (возможно, это покажется странным, но, если не готовиться к занятию заранее, то ничего не успеешь и ничего не получится).

Остались вопросы? Используй второе правило!

Если с этим закончили, то перейдем к чисто технической части - подготовим к работе рабочее место и все настроим.

Эта работа будет разделена на 2 части:



- Подготовку самого микроконтроллера (вот этой штуки:)
- И настройку IDE (integrated development environmen) - **Интегрированной среды разработки** для его программирования.

Подготовка микроконтроллера

Что такое микроконтроллер? Что он может? Зачем он нужен? Все это мы подробно разберем на первой лекции, а пока, упрощенно, можно представить его как очень простой и дешевый компьютер.

Без чего не может работать компьютер? Без операционной системы.

Это очень грубое приближение, и, говоря по правде, не совсем верное, но в начале нашего пути вполне допустимое для старта.

В нашем курсе мы будем работать с микроконтроллером [Raspberry Pi Pico](#). Он довольно универсален и позволяет писать для себя программы на двух языках программирования:

- [C/C++](#)
- [Micropython](#)

Программы на `C/C++` работают быстрее, но маркшейдер по своей природе терпелив и рационален. Наш основной курс "Алгоритмы и программы автоматизации..." основан на `Python`, поэтому выбор `Micropython` для этого курса более предпочтителен.

В чем отличия `Micropython` от `Python`? Сторонний наблюдатель издали не заметит разницы. Общий синтаксис и правила языка одинаковы (есть различия в частностях, но на них мы будем останавливаться по мере их возникновения).

В целом `Micropython` это просто максимально упрощенный (в плане наворотов и дополнительных пакетов) версия обычного `Python`. Это сделано для экономии скромных ресурсов микроконтроллера по памяти и вычислительной мощности. Дополнительно в `Micropython` встроен модуль `machine`, которого нет в старшем брате языка. Этот модуль нужен для непосредственной работы с микроконтроллером и управления платой.

Но хватит слов - пора за дело!

Прошивка платы

Важный дисклеймер:

⚠ Прошивку платы нужно сделать всего ОДИН раз (не нужно с этого начинать каждое занятие!). *Исключение* составляет только, если вы хотите обновить версию `micropython`, которые выходят с периодичностью 1-2 раза в год.

Процесс прошивки микроконтроллера показан в [туториале по запуску](#) (с видео, но на английском).

Но если коротко:

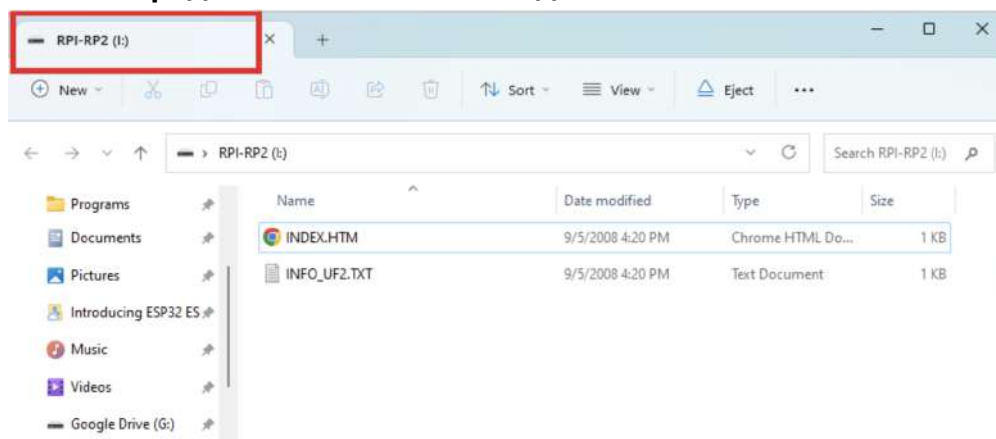
1. Скачиваем загрузчик языка:

- Страница загрузчиков для [Raspberry Pi Pico](#)
- [Резервная ссылка на диск \(v1.26.0\)](#)

2. Подключаем к компьютеру микроконтроллер по кабелю с зажатой кнопкой BOOTSEL (Единственная механическая кнопка на плате).



3. Плата определится как внешний диск.



4. Скидываем файл прошивки на плату (она перезапустится и перестанет определяться).

5. Плата готова к работе!

Подготовка IDE

На сегодняшний день есть две основных IDE для программирования на `Python`:

- [PyCharm](#)
- [VS Code](#)


Что выбрать?

PyCharm - это, фактически, единственная IDE, разработанная для программирования на `Python`. Она активно развивается, поддерживается - в целом она практически идеальна для работы и обучения. Базовым преимуществом PyCharm является то, что она работает сразу "из коробки" и содержит все необходимые инструменты сразу.

VS Code - универсальная IDE - в ней можно программировать на любых языках, но она требует ручной "донастройки" путем установки сторонних плагинов. Это не является критическим недостатком, но ухудшает общую унификацию этой программы между разработчиками.

Кажется, что PyCharm лучше... Тем не менее, ответ на вопрос, где конкретно работать не имеет однозначного ответа. В прошлом 2024 году, как и в предыдущие, оптимальным был PyCharm (именно он и установлен в наших компьютерных классах). Принципиально для разработки приложений на базовом `Python` ничего не изменилось и PyCharm стоит попробовать и использовать.

Однако совокупность мелких недостатков и накопленных противоречий диалектически перешло, к настоящему времени, в качество и, опуская скучные перечисления, следует признать, что **конкретно для обучения программированию на RP Pico VS Code лучше.**

 Мы разберем установку и настройку обеих программ, в целом окончательный выбор за Вами!


VS Code

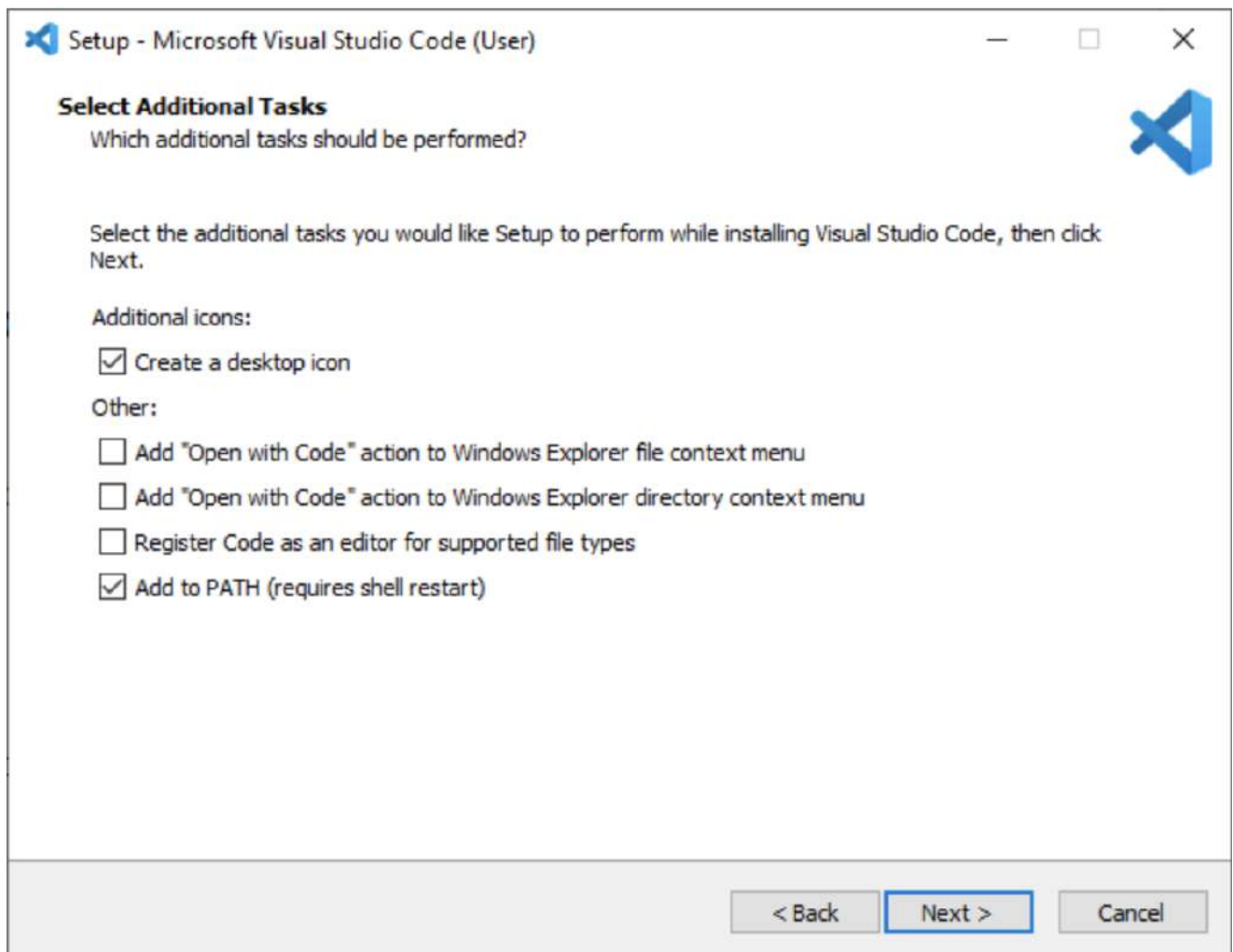
Скачиваем и устанавливаем VS Code

Тут нет никаких сложностей и подводных. Переходим на официальный [сайт программы](#), скачиваем дистрибутив под свою систему и устанавливаем программу стандартным способом.

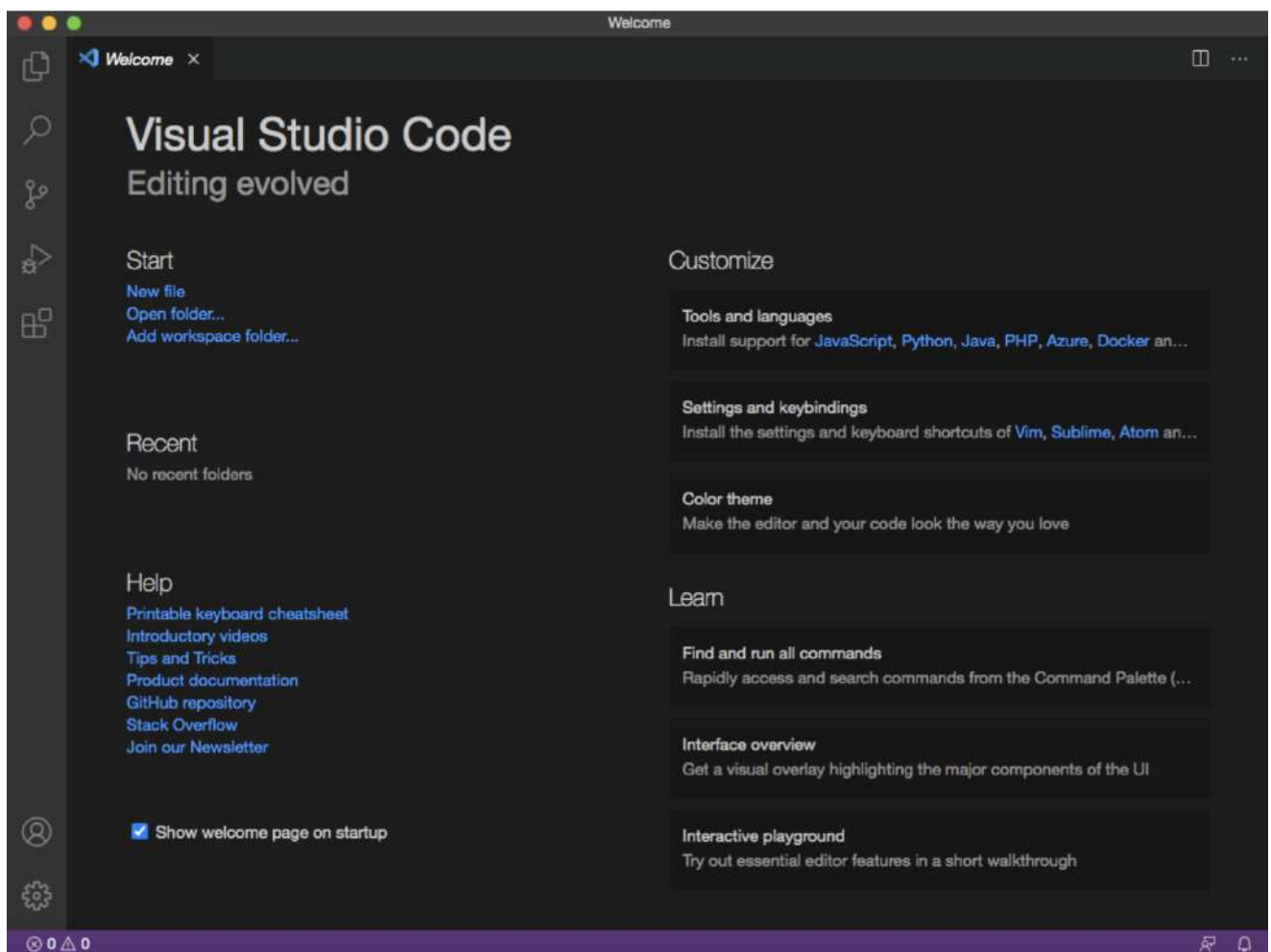
Дистрибутивы актуальных версий (на 31.08.2025) можно также скачать с диска:

- [Для Windows](#)
- [Для MacOS](#)

 Это не обязательно, но, для более удобного открытия файлов из системы Windows, можно выбрать все чек боксы из списка при установке:



После успешной установки программа будет выглядеть примерно так:

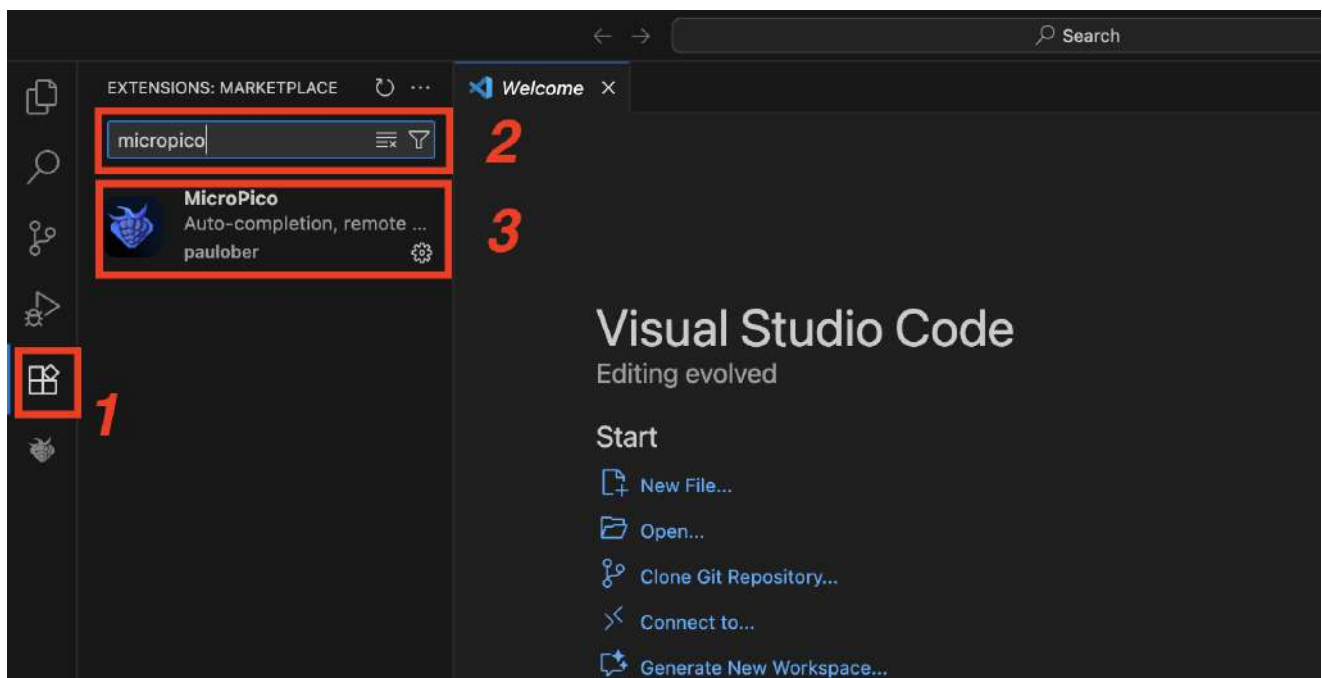


Сейчас это просто текстовый редактор - мы не сможем запустить из него ни какой код.

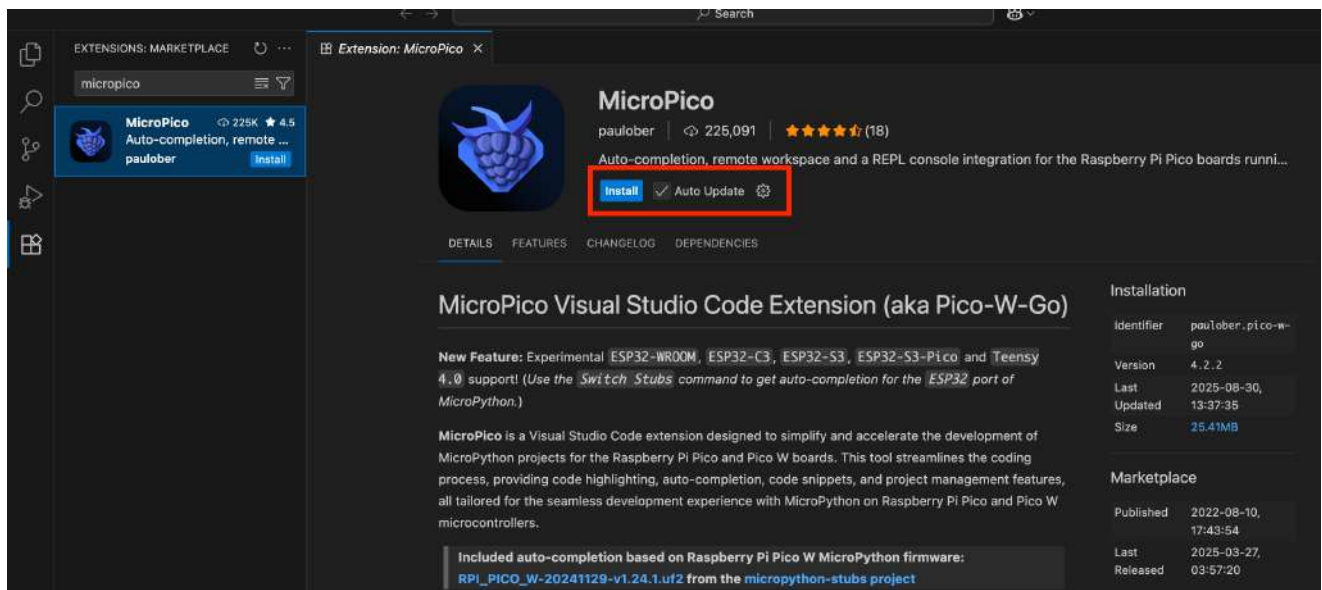
Установка плагина для `micropython`

Для работы в VS Code необходимо скачать дополнительные плагины, для этого:

1. Выбираем в левой панели вкладку "Extensions"
2. В поле поиска набираем название плагина `micropico` (или старое `pico-w-go`)
3. Выбираем плагин как на картинке:



4. И нажимаем кнопку "установить":



После этого плагин автоматически скачается со всеми необходимыми зависимостями (дополнительными плагинами).

Очевидно, что без интернета это не сработает...

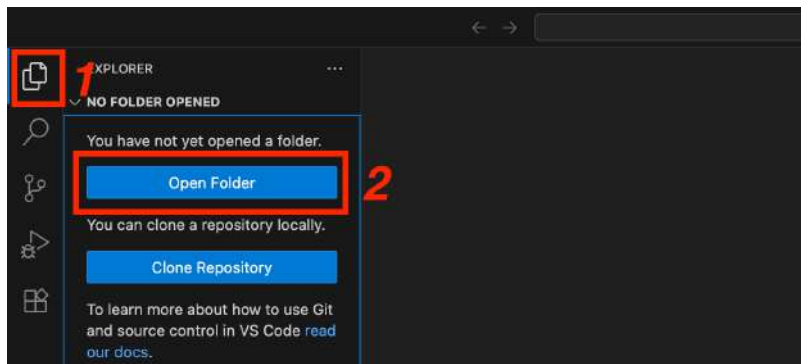
Инициализация проекта

Когда все установлено необходимо создать и инициировать проект для работы с микроконтроллером.

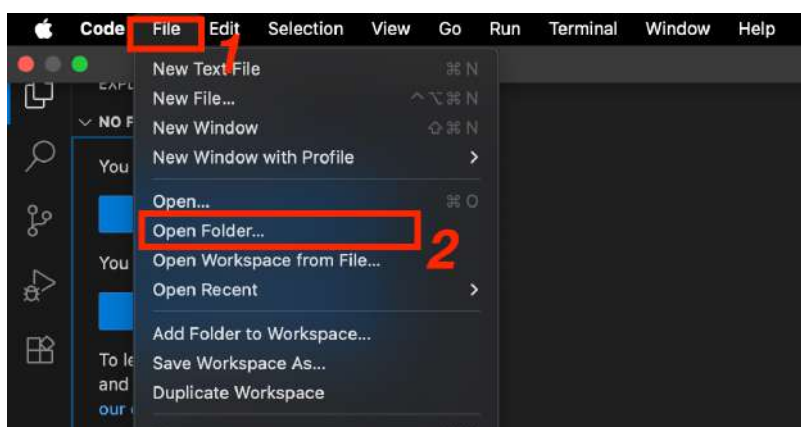
⚠ Нет никакой необходимости создавать каждый раз новый проект! Наоборот, весь смысл проекта в том, что он обобщает в себе большой объем кода. **Лучше выполняйте весь курс в одном проекте, если отдельно не будет сказано обратного!**

Для создания проекта для `micropython` в `VS Code` необходимо:

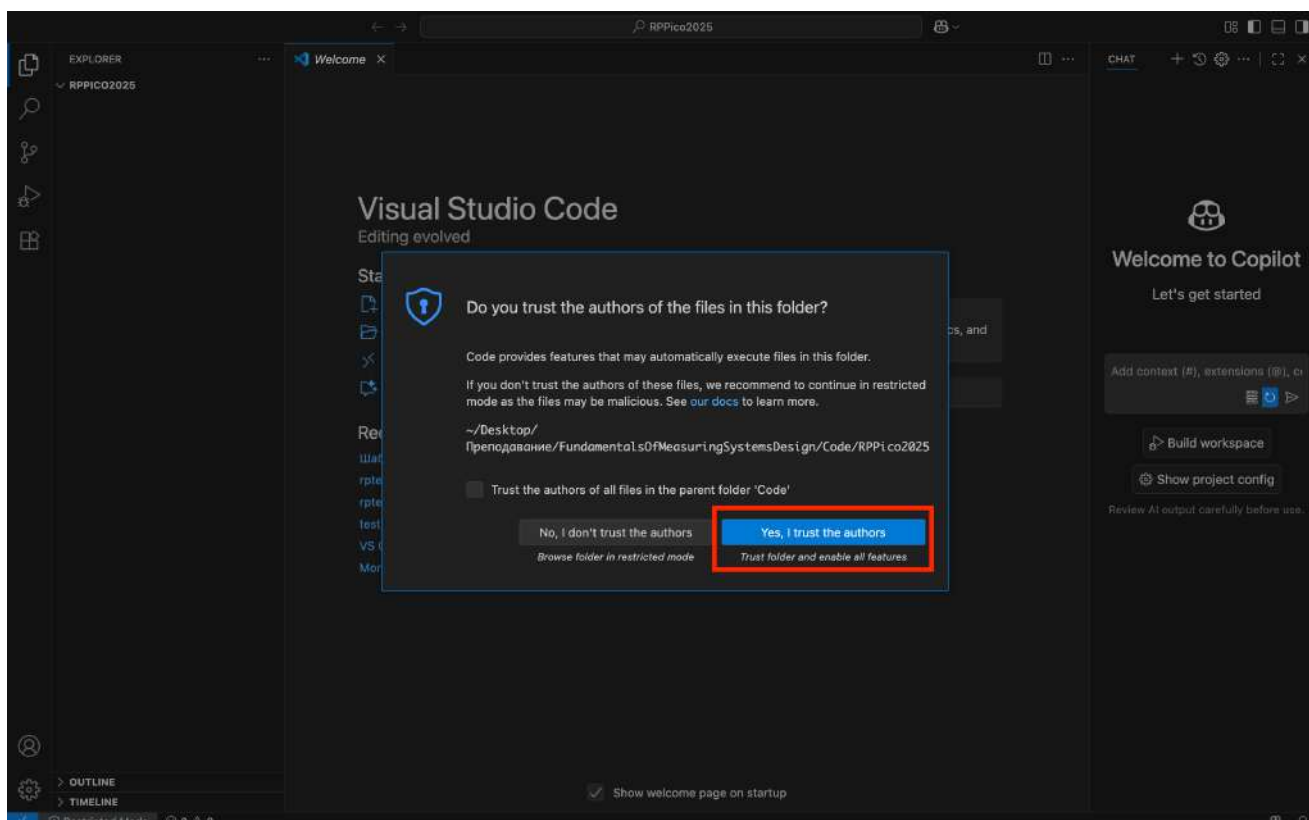
1. Создаете обычную папку для проекта в удобном Вам месте.
2. В панели `Explorer` выберите кнопку "Open Folder"



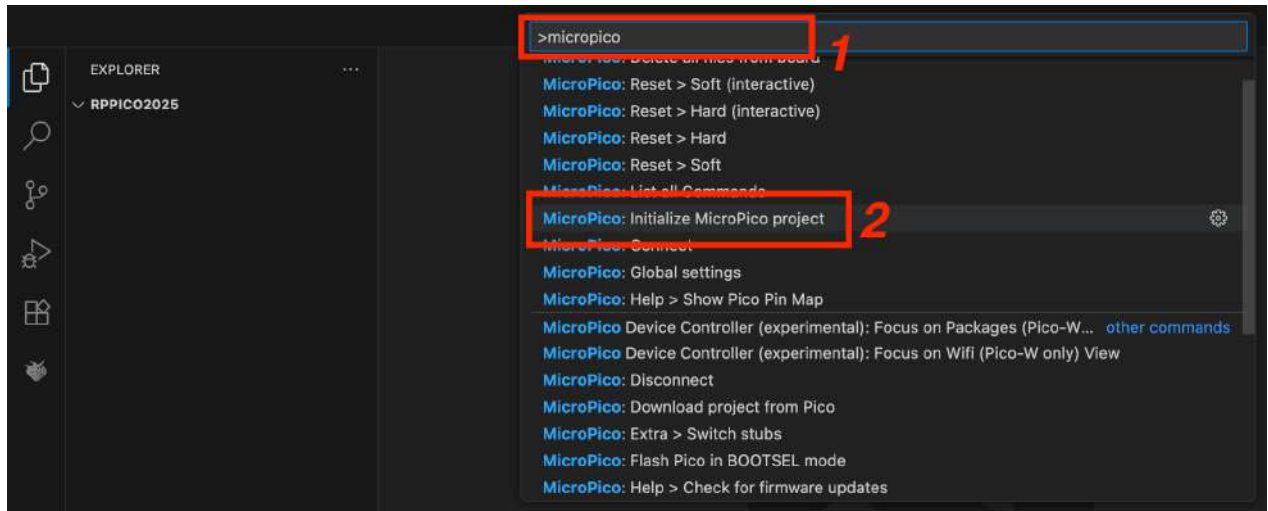
или выберите "Open Folder..." в меню "file":



3. Выберите ранее созданную папку в меню и подтвердите, что "доверяете" создателю папки:



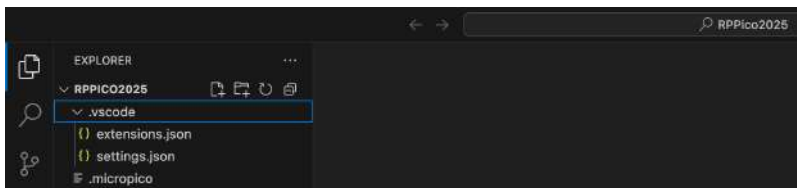
4. Нажмите комбинацию клавиш **Ctrl + Shift + P** и в появившемся окне наберите название плагина: **micropico**. В выпавшем списке выберите "Initialize MicroPico project":



В случае успеха внизу экрана появится сообщение о завершении конфигурации проекта, а так же появится меню плагина:



В самой папке проекта будут созданы скрытые файлы с необходимыми настройками:



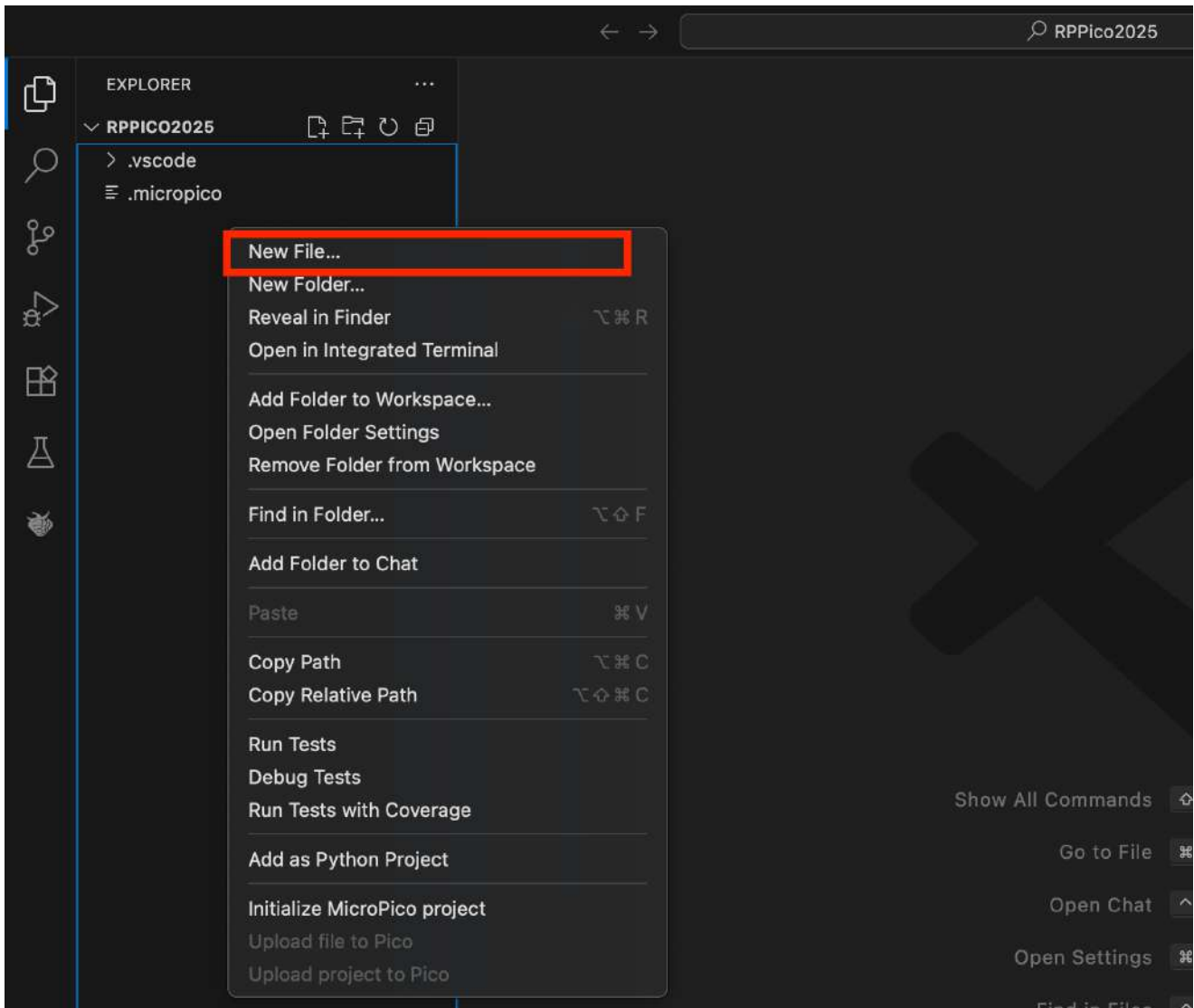
Не удаляйте их!

Первая программа

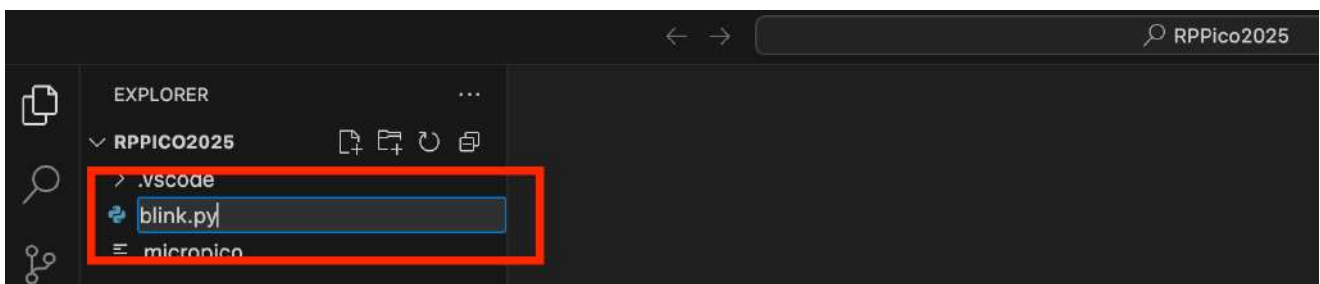
Написание программы

Для начала создадим новый питоновский файл (файл с расширением **.py**) в папке с проектом. Для этого щелкаем в проводнике правой кнопкой мыши (ПКМ) и выбираем

"New File...":



и называем его в соответствии с будущим содержанием программы:



- blink - "моргание" стандартный тест работоспособности платы, предполагающий последовательное включение-выключение встроенного в плату светодиода;
- **обязательно допишите в конце названия файла `.py` !**

Запишем в файл следующий код:

```
from machine import Pin
import time

led_pin = Pin("LED", Pin.OUT)
```

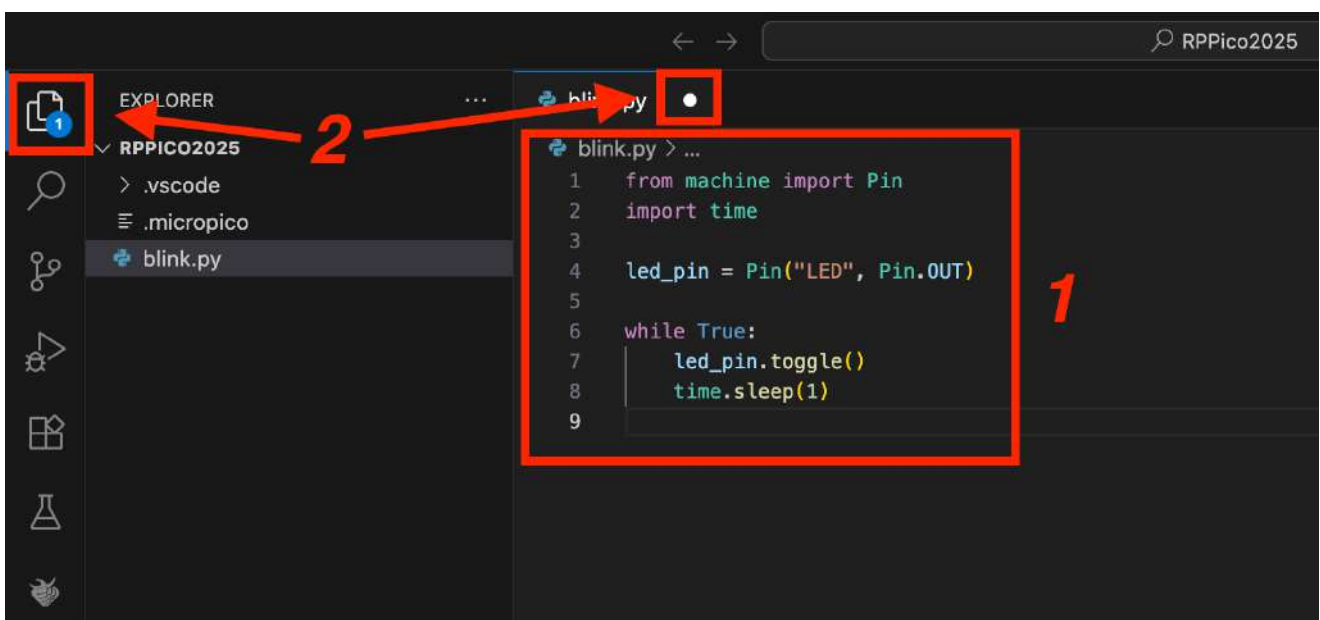
```
while True:
    led_pin.toggle()
    time.sleep(1)
```

Пока не будем подробно разбирать принцип его работы, но синтаксис `python` довольно интуитивно понятен и человекочитаем, поэтому попробуйте понять основной смысл кода самостоятельно!

⚠ Отступы внутри блоков кода важны! Стандартом являются 4 пробела (ставятся автоматически кнопкой табуляции)!

VS Code автоматически не сохраняет изменения в файлах!

Поэтому набрав код (1), заметьте, что во вкладке с файлом появилась белая точка, а в кнопке проводника - счетчик с количеством измененных файлов:

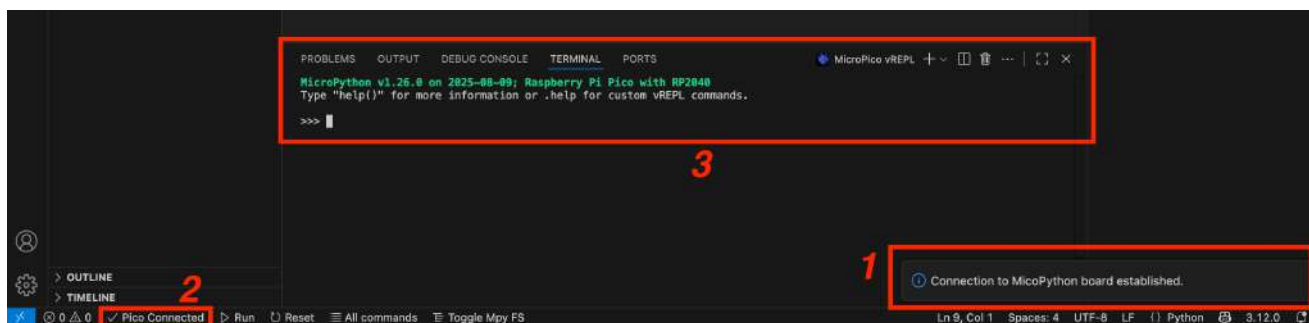


Сохраните изменения в файле нажав `Cntrl + S` и эти знаки должны пропасть.

Запуск программы

Когда программа готова, подключаем микроконтроллер к компьютеру.

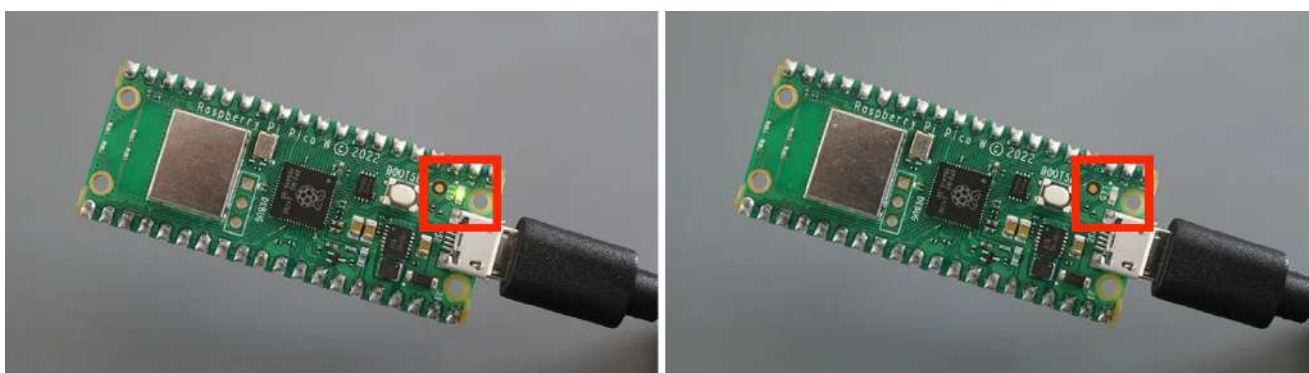
VS Code должен самостоятельно определить плату и подключиться к ней. В случае успеха внизу появится сообщение об обнаружении платы (1) и изменится состояние подключения (2). Кроме того откроется интерактивный REPL терминал (3) в котором можно передавать команды на плату напрямую:



Запустить выполнение программы из текущего файла на микроконтроллере можно, щелкнув на "Run" в нижней панели управления:



Если все было сделано верно, диод на плате начнет мигать с частотой в 1 секунду:

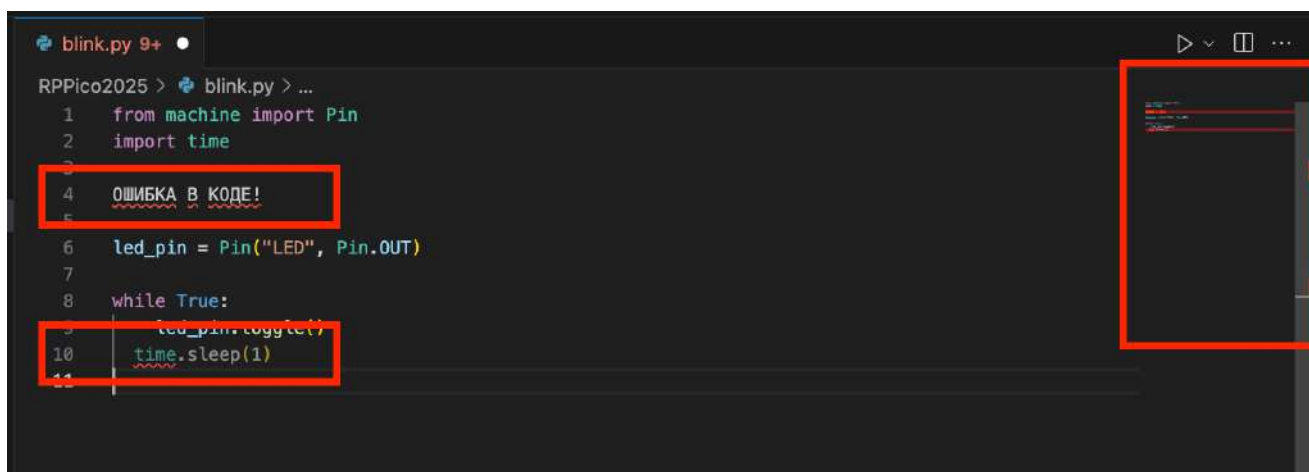


Остановить выполнение программы можно нажав на "Stop":



Отладка ошибок

Предположим, что при написании программы вы допустили некоторые ошибки:



Здесь в 4 строке мы допустили синтаксическую ошибку набрав несколько имен переменных подряд и не присвоив им никаких значений. А в 10 строке поставили неправильный отступ (2 пробела вместо 4).

❗ Синтаксические ошибки легко исправить и они автоматически выделяются в IDE. С логическими все несколько сложнее...

При запуске программы она, очевидно, не сможет работать, а в терминале появится ошибка с указанием типа и строки, где она возникла:

```
PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS MicroPico vREPL + - [ ] [ ] ... [ ] [ ] X
MicroPython v1.26.0 on 2025-08-09; Raspberry Pi Pico with RP2040
Type "help()" for more information or .help for custom vREPL commands.

>>>
Traceback (most recent call last):
  File "<stdin>", line 4
SyntaxError: invalid syntax

>>> █
```

Фактически программа "сломается" на первой ошибке, не дойдя до второй. Поэтому при повторном перезапуске (после исправления первой ошибки) мы получим следующее сообщение о неправильном отступе:

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS MicroPico vREPL + - [ ] [ ] ... [ ] [ ] X
MicroPython v1.26.0 on 2025-08-09; Raspberry Pi Pico with RP2040
Type "help()" for more information or .help for custom vREPL commands.

>>>
Traceback (most recent call last):
  File "<stdin>", line 4
SyntaxError: invalid syntax

>>>
Traceback (most recent call last):
  File "<stdin>", line 9
IndentationError: unindent doesn't match any outer indent level

>>> █
```

❗ Хотя ошибка и в 10 строке - программа "сломается" при переходе к ней на 9 поэтому в указании локализации ошибки может быть небольшое смещение.

После исправления ошибки программа должна запускаться и работать штатно.

Запись программы на микроконтроллер

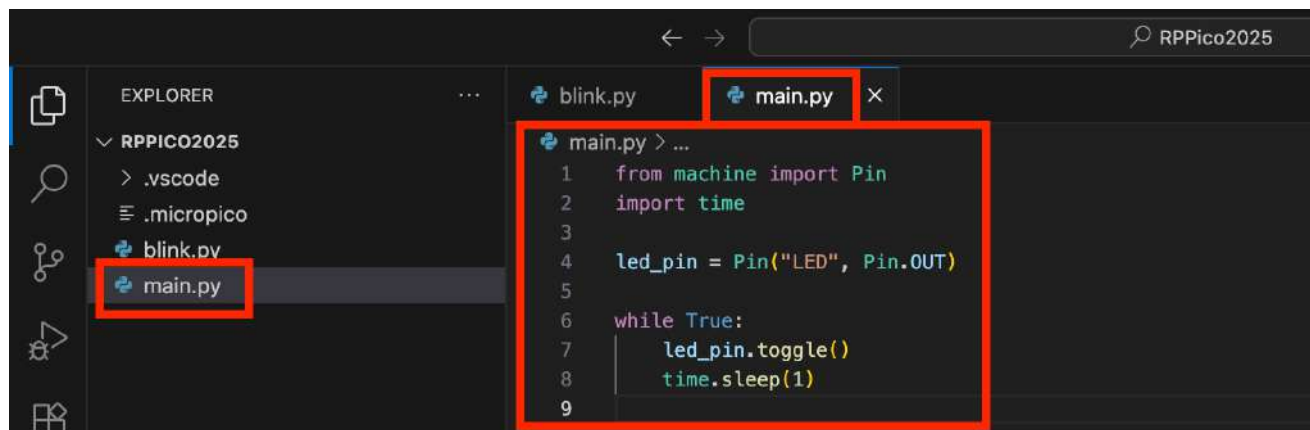
Хотя микроконтроллер и выполнял нашу программу (она выполнялась на его процессоре) сам код не был записан в его память, а значит без VS Code и внешнего компьютера он выполняться не будет.

Для создания независимого устройства необходимо сохранить код непосредственно на плату.

До этого мы запускали код из текущего выбранного файла, поэтому не было принципиально как он называется. Однако когда плата будет работать сама по себе так делать уже не получится - микроконтроллер "не поймет" откуда ему начинать работу!

⚠ Микроконтроллер RP Pico ВСЕГДА запускает файл с названием `main.py` !
Именно от является точкой входа и запуска программы!

Для того чтобы плата мигала диодом независимо от внешнего компьютера создадим новый файл `main.py` и скопируем в него ранее проверенный код:

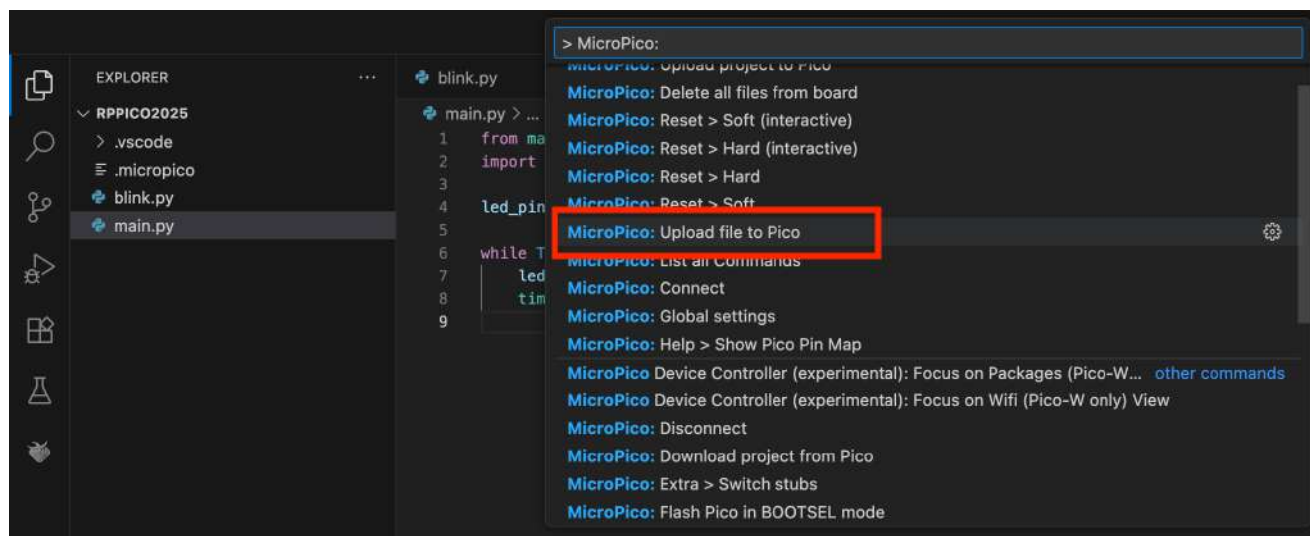


В нижней панели можно выбрать список всех доступных команд для управления микроконтроллером:



ℹ Кнопка работает аналогично ранее использованной комбинации `Ctrl + Shift + P` с указанием плагина

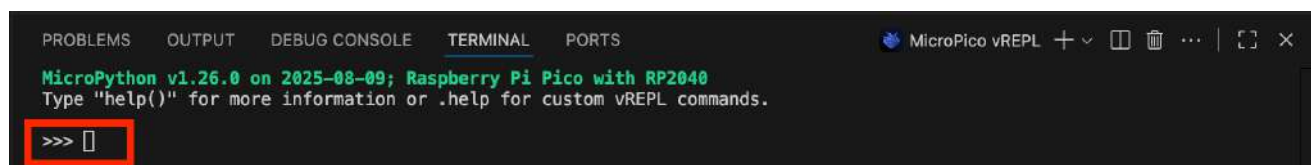
В появившемся списке выбираем "Загрузить файл на плату":



⚠ Для загрузки файла необходимо свободное соединение с платой. Если при загрузке файла запущено выполнение кода с компьютера загрузка не произойдет.
Предварительно нужно остановить выполнение текущей программы!

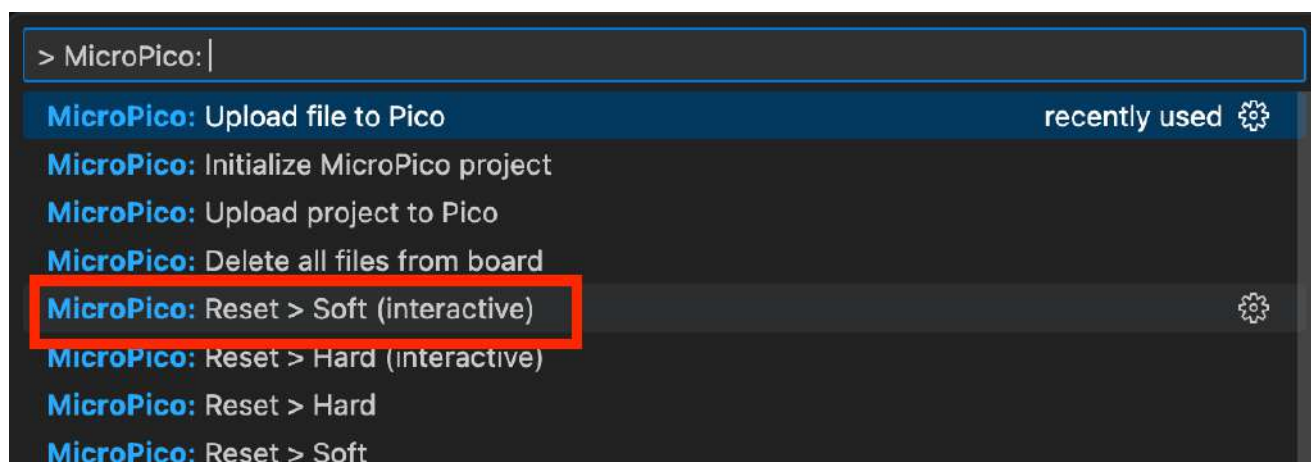
Если все пройдет нормально, внизу появится сообщение об успешной загрузке файла. Однако ожидаемого моргания диода не начнется.

Причина в том, что после загрузки файла плата автоматически перезагрузится и подключенный к компьютеру микроконтроллер перейдет в режим ожидания команд с терминала. Визуально это видно по указателю из трех стрелочек:

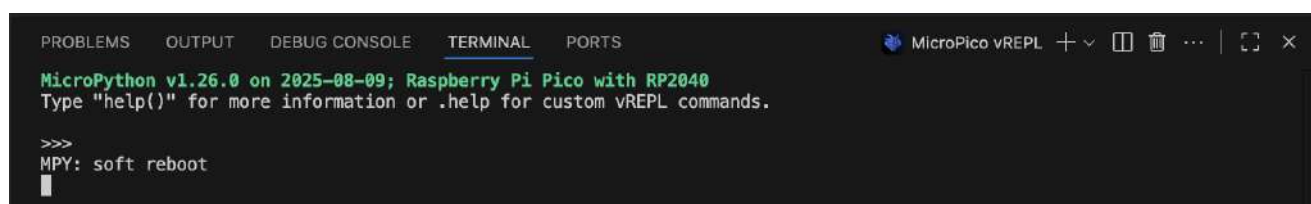


В режиме ожидания вся работа микроконтроллера приостанавливается до получения новых инструкций.

Выйти из режима ожидания можно, перезагрузив микроконтроллер в специальном режиме:



После этого плата начнет работать согласно записанной на нее программы, а в терминале не появится указатель интерактивного ввода команды:



❗ После записи программы в память микроконтроллера внешний компьютер теряет необходимость. Вы можете подключить плату к любому внешнему источнику питания и она будет работать согласно программе.

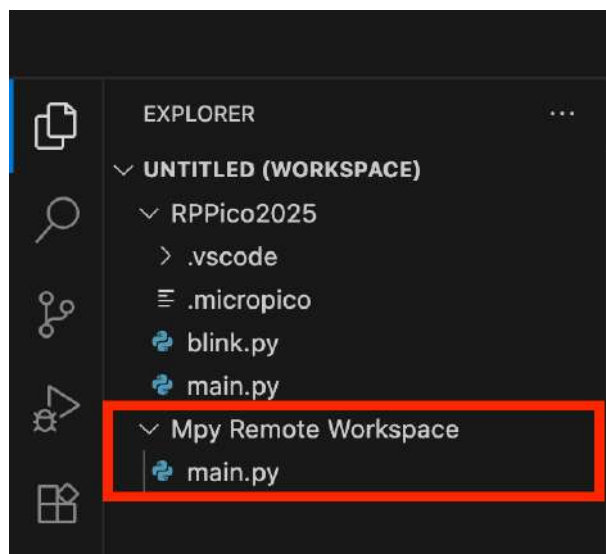
Управление файлами на микроконтроллере

В плагине для VS Code реализован удобный инструмент работы с данными на микроконтроллере.

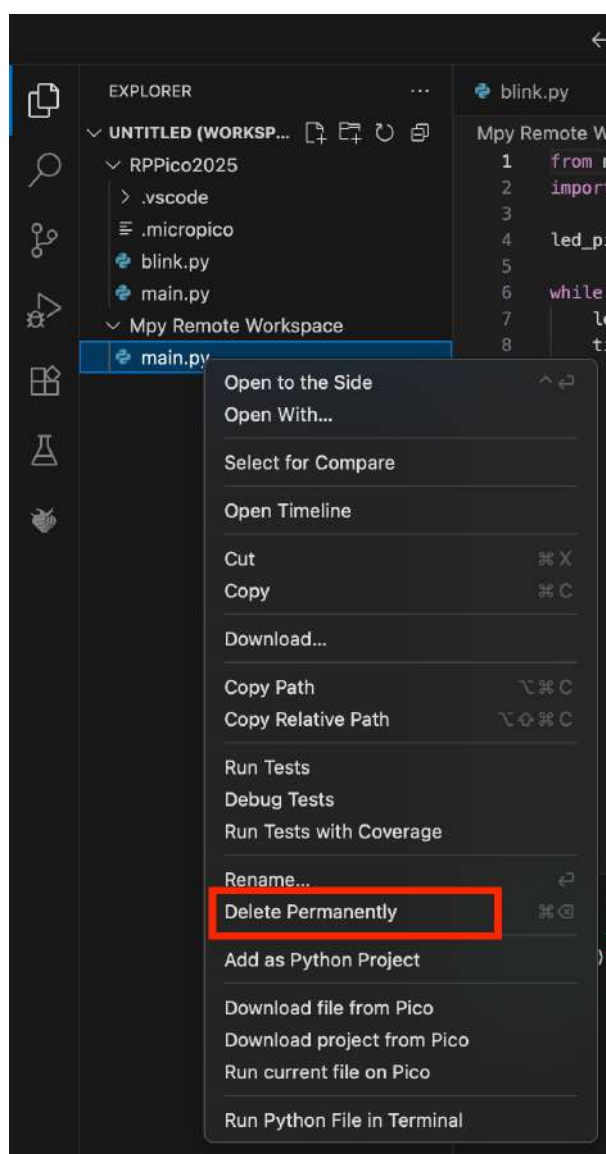
Для его включения необходимо нажать кнопку:



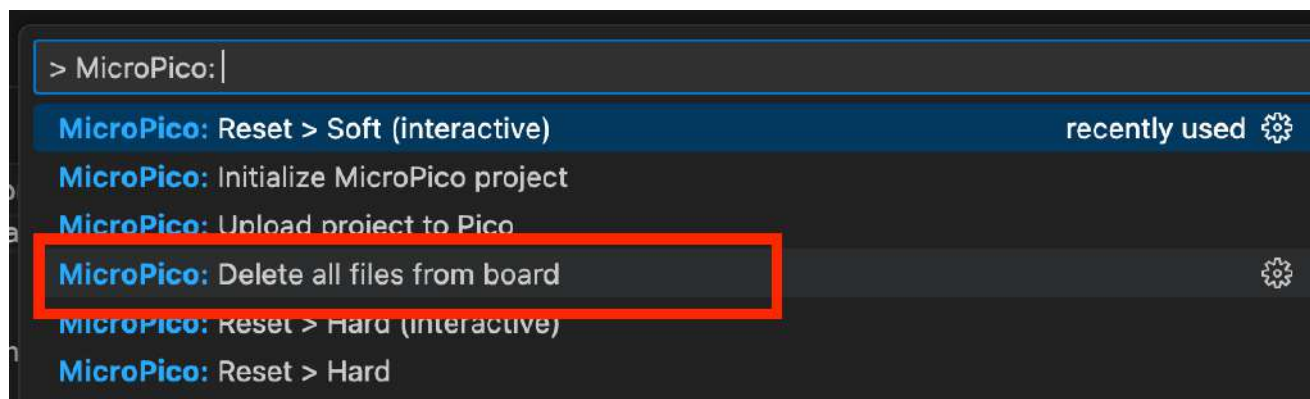
После чего в проводнике проекта появится новое меню, относящееся к файлам записанным на плату:



В этом режиме можно редактировать содержимое файла, редактировать его название, скачивать его в память компьютера и удалять:



Очистить память микроконтроллера можно выбрав соответствующую команду из списка:



⚠ Так как с одной и той же платой могут работать разные люди и выполнять разные задания, удаляйте все файлы по окончании занятия!

PyCharm

Скачиваем и устанавливаем PyCharm

PyCharm - мощная и удобная IDE, до 2025 года она имела 2 версии:

- Платную PyCharm Pro (с 30 дневным пробным периодом);
- И бесплатную PyCharm Community Edition.

Сейчас версия одна и она бесплатная, то что было в платной версии доступно по подписке.

Бесплатного функционала нам с избытком достаточно.

Скачать текущую версию программы можно с официального [сайта](#).

Однако не спешите пытаться ее скачивать.

Компания JetBrains внезапно (в течении последних месяцев) забыла свои русские корни и ограничила доступ к скачиванию своих продуктов с территории РФ.

📄 Через VPN все скачивается и устанавливается без проблем.

Для Вашего удобства я скачал все необходимые дистрибутивы на диск:

PyCharm 2025.2.1

- [Ссылка для Windows](#)
- [Ссылка на MacOS \(чип Intel\)](#)

Это последняя на текущий момент (31.08.2025) версия.

⚠ Эта версия программы может быть Вами использована в основном курсе по программированию, однако она не подходит для работы с `micropython`, так как плагин для работы с ним перестал поддерживать версии выше 2024.2.6

Скачать такую версию можно по ссылкам,

PyCharm 2024.2.6 Community Edition

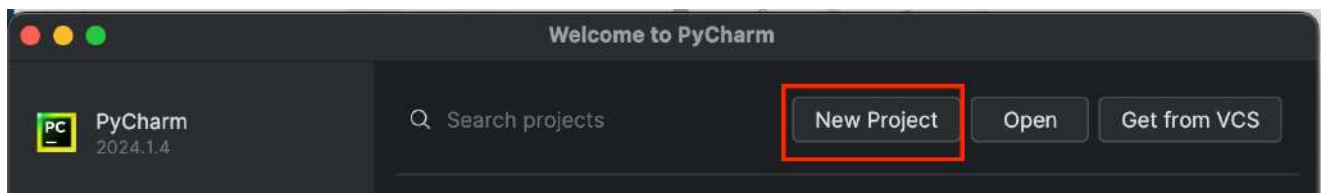
- [Ссылка для Windows](#)
- [Ссылка на MacOS \(чип Intel\)](#)

После того, как Вы скачали дистрибутив, он устанавливается стандартным способом.

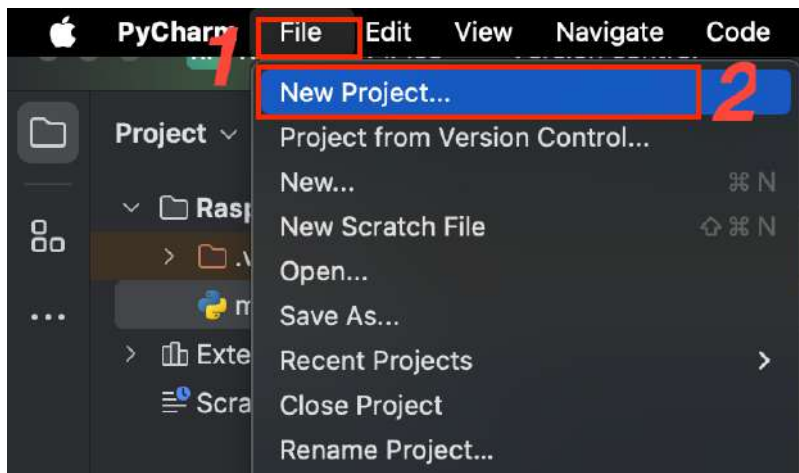
Создаем проект для разработки

i Все, что было справедливо к требованиям для VS Code справедливо и для PyCharm!

Нажмите "Новый проект" при запуске программы:



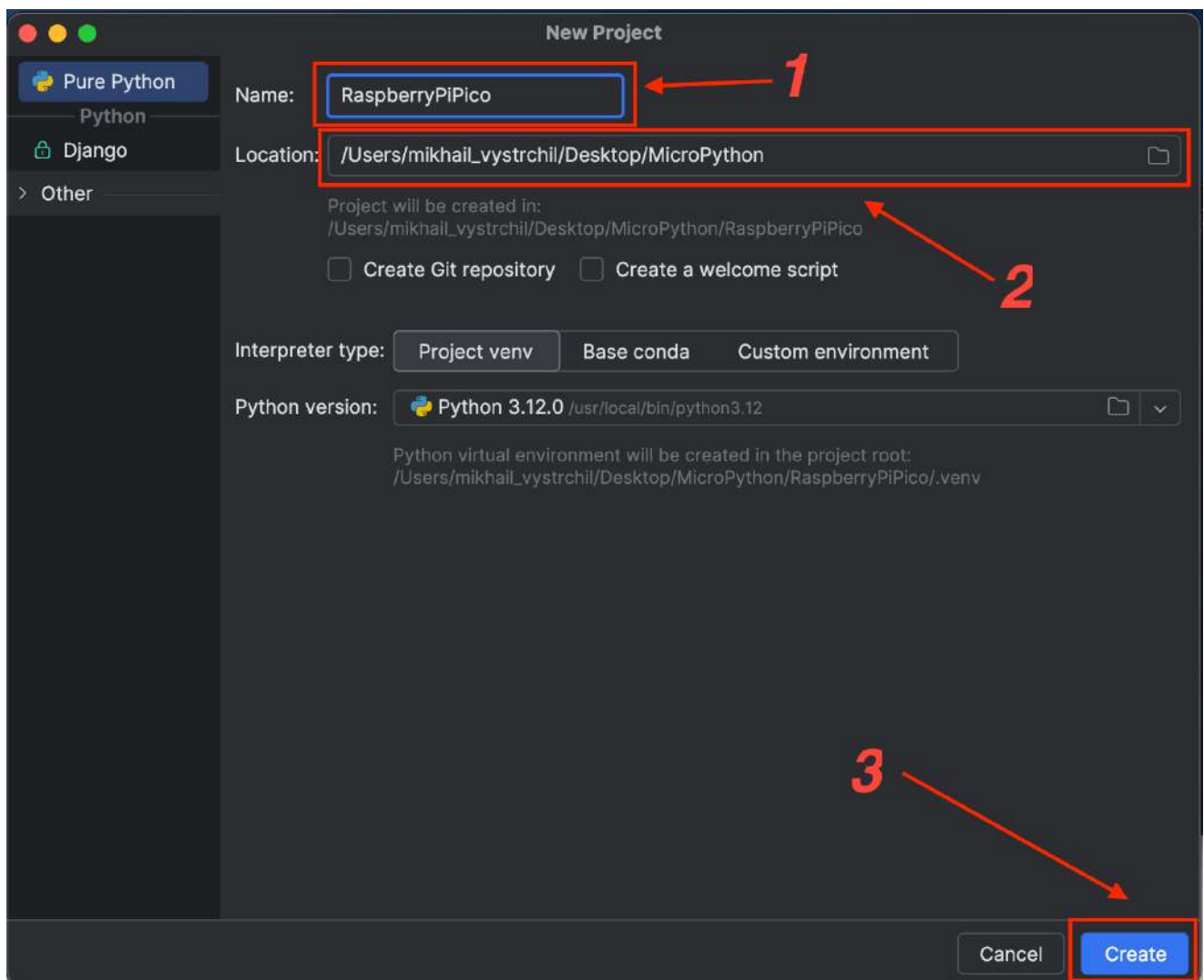
Или выберите "Новый проект" в меню "файл":



В поле 1 пишется Ваше оригинальное название проекта (постарайтесь придумать что-нибудь получше чем я :))

В поле 2 задается путь к той директории в которой будет создан проект.

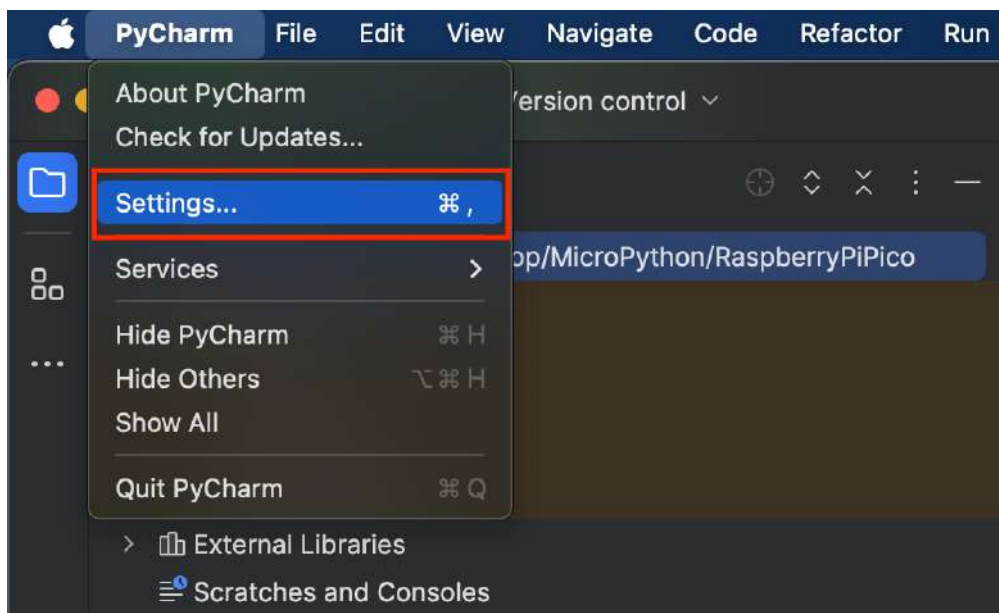
Обратите внимание, что сам проект - это, по сути папка, в которой будет формироваться структурированный код программы и именно папка с названием проекта (с дополнительными служебными файлами) и будет создана в указанной в поле 2 директории.



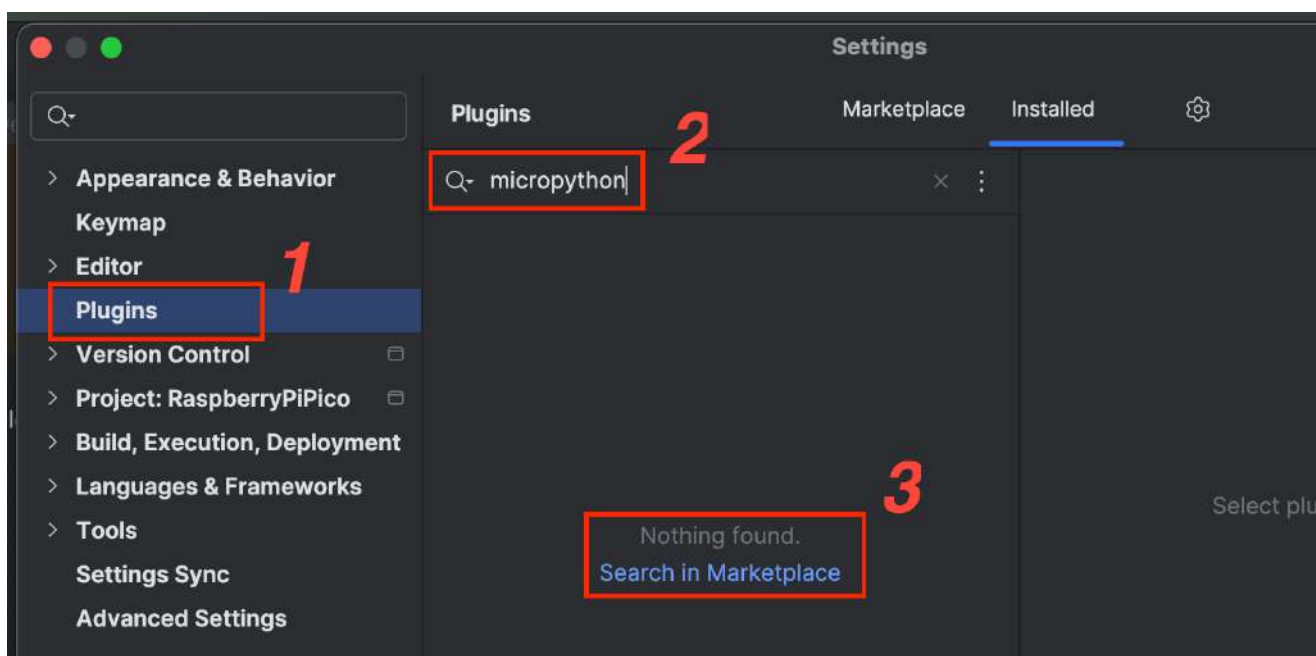
⚠ Нет никакой необходимости создавать каждый раз новый проект! Наоборот, весь смысл проекта в том, что он обобщает в себе большой объем кода. **Лучше выполняйте весь курс в одном проекте, если отдельно не будет сказано обратного!**

Установка и настройка плагина MicroPython

Для начала работы с микроконтроллером необходимо предварительно установить в PyCharm дополнительный плагин "Micropython". Для этого необходимо зайти в меню "настройки":

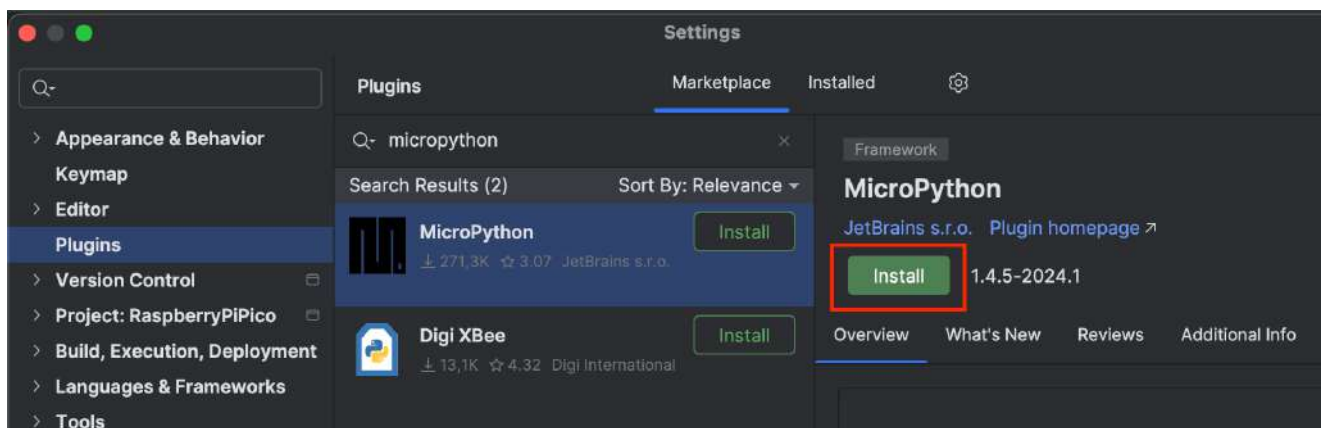


В открывшемся окне выбрать вкладку 1 "плагины", ввести в поиске 2 "micropython" и выбрать опцию 3 запускающую поиск в интернете.

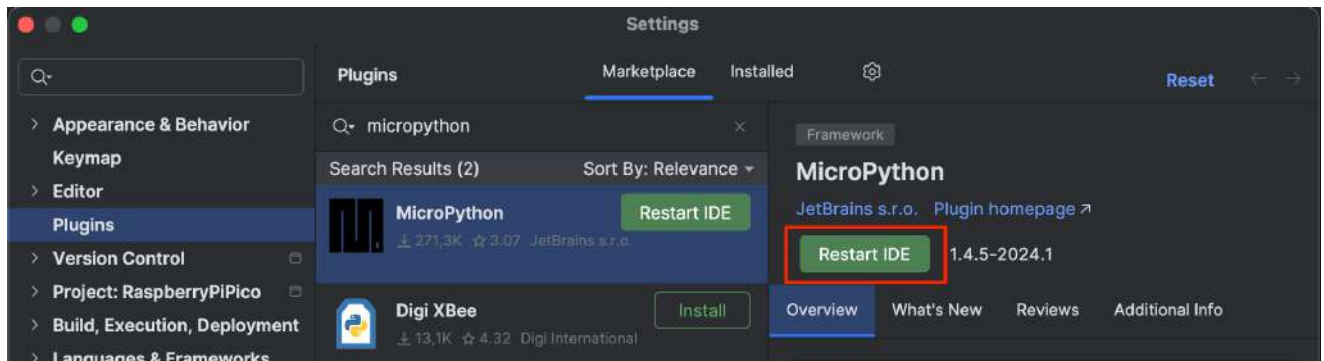


❗ В условиях "Горного университета" из-за доступа в интернет через проху сервер поиск плагина возможен только после выполнения настроек сервера. Ниже описано как установить плагин вручную.

После чего необходимо нажать кнопку "установить" для нужного плагина:



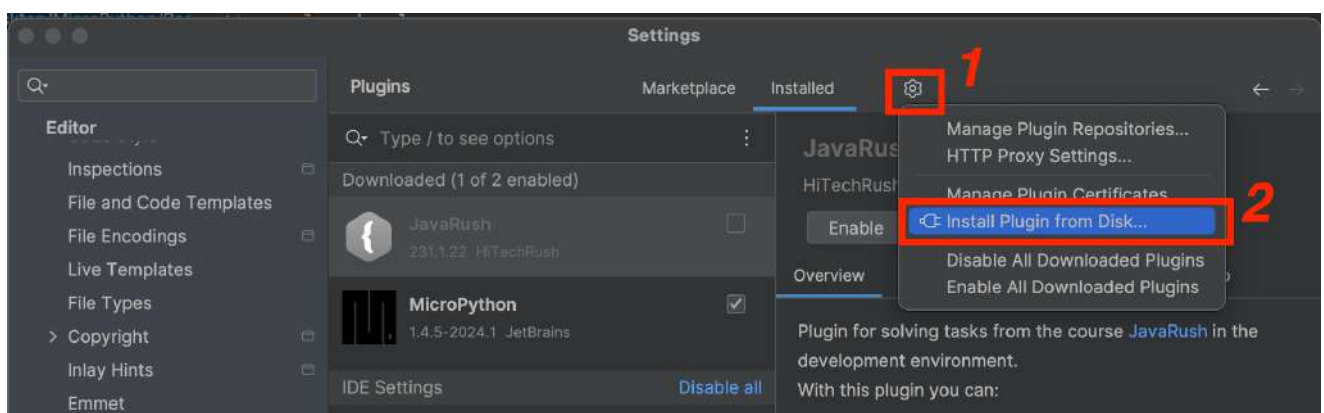
И после установки перезапустить PyCharm:



⚠ Без VPN установка плагинов стандартным способом может быть ограничена!

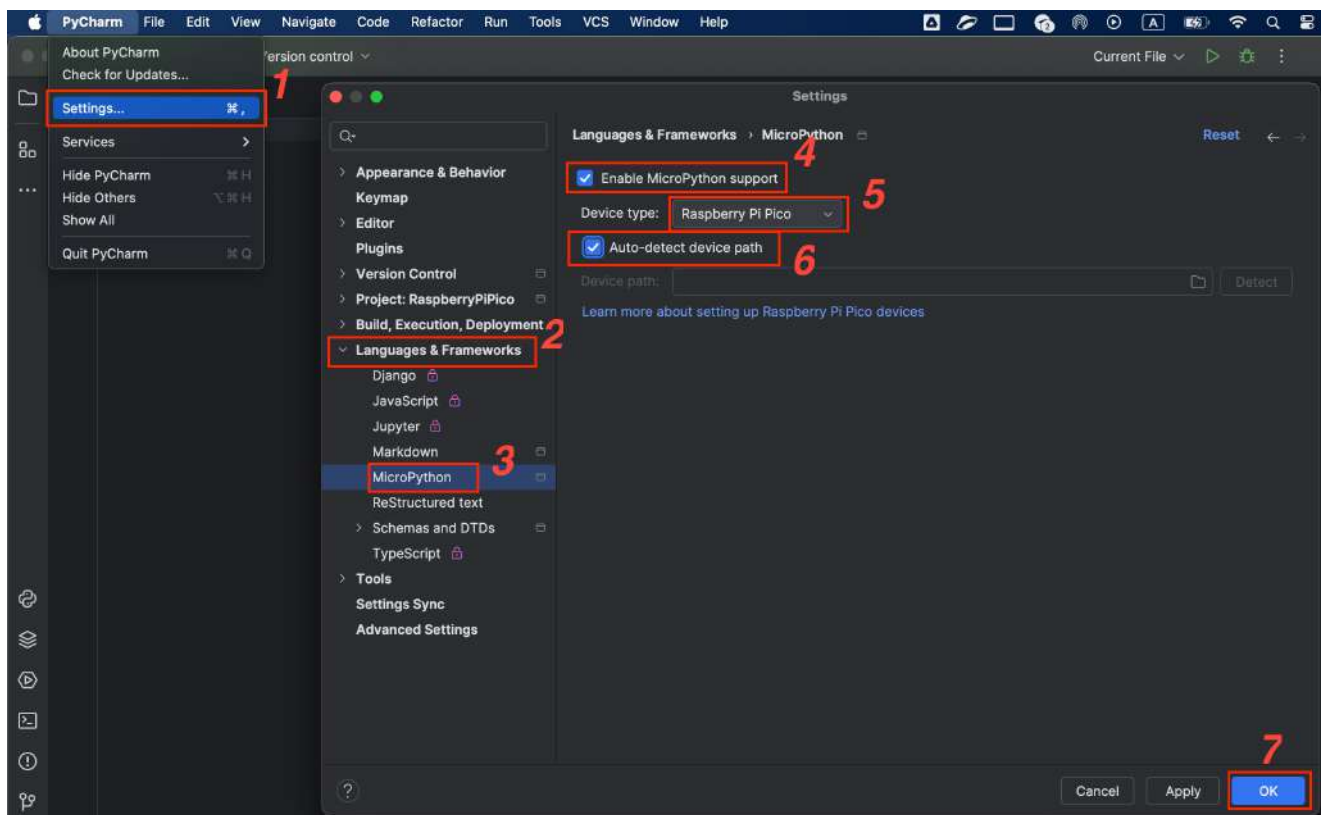
Для установки плагина вручную скачайте архив с ним с [сайта](#) или с [диска](#)

В меню установки плагинов нужно нажать на шестеренку (1) и выбрать установку плагина с диска (2):

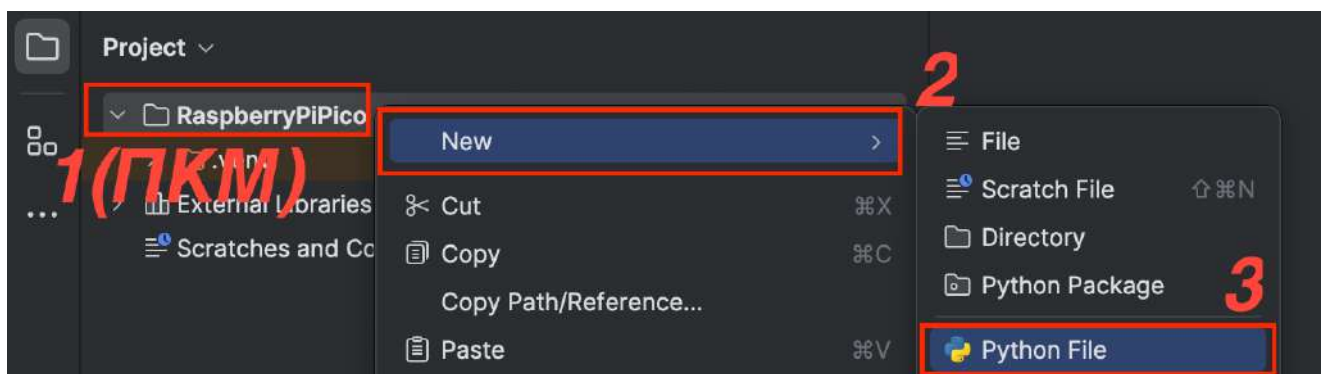


После чего выбрать скачанный архив с плагином. После установки необходимо перезапустить PyCharm.

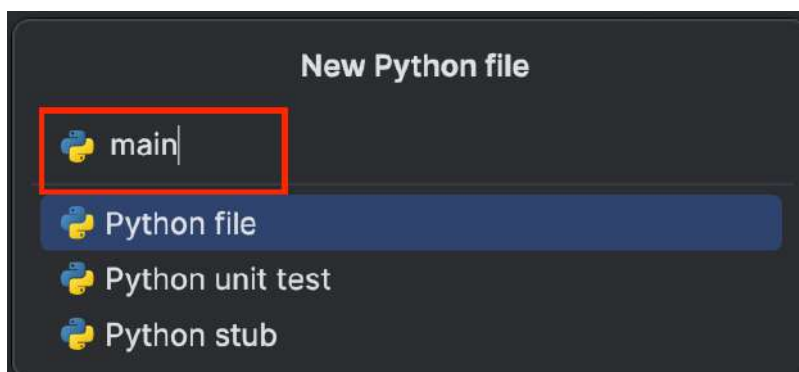
После перезапуска PyCharm в меню (1) "Настройки", вкладке (2) "Языки и Фреймворки" появится язык MicroPython. Выбрав поле (3) нужно поставить галочку в поле (4) и подключив тем самым сам язык в текущий проект. Далее необходимо выбрать тип используемого микроконтроллера (Raspberry Pi Pico в поле (6)). И выбрать авто определение подключенного устройства в поле (6), после чего нажать "применить" и "ОК" (7):



Далее создадим в проекте новый `.py` файл в котором начнем набирать код:



Обратите внимание, что название создаваемого файла обязательно должно быть **main** (расширение файла добавится само):

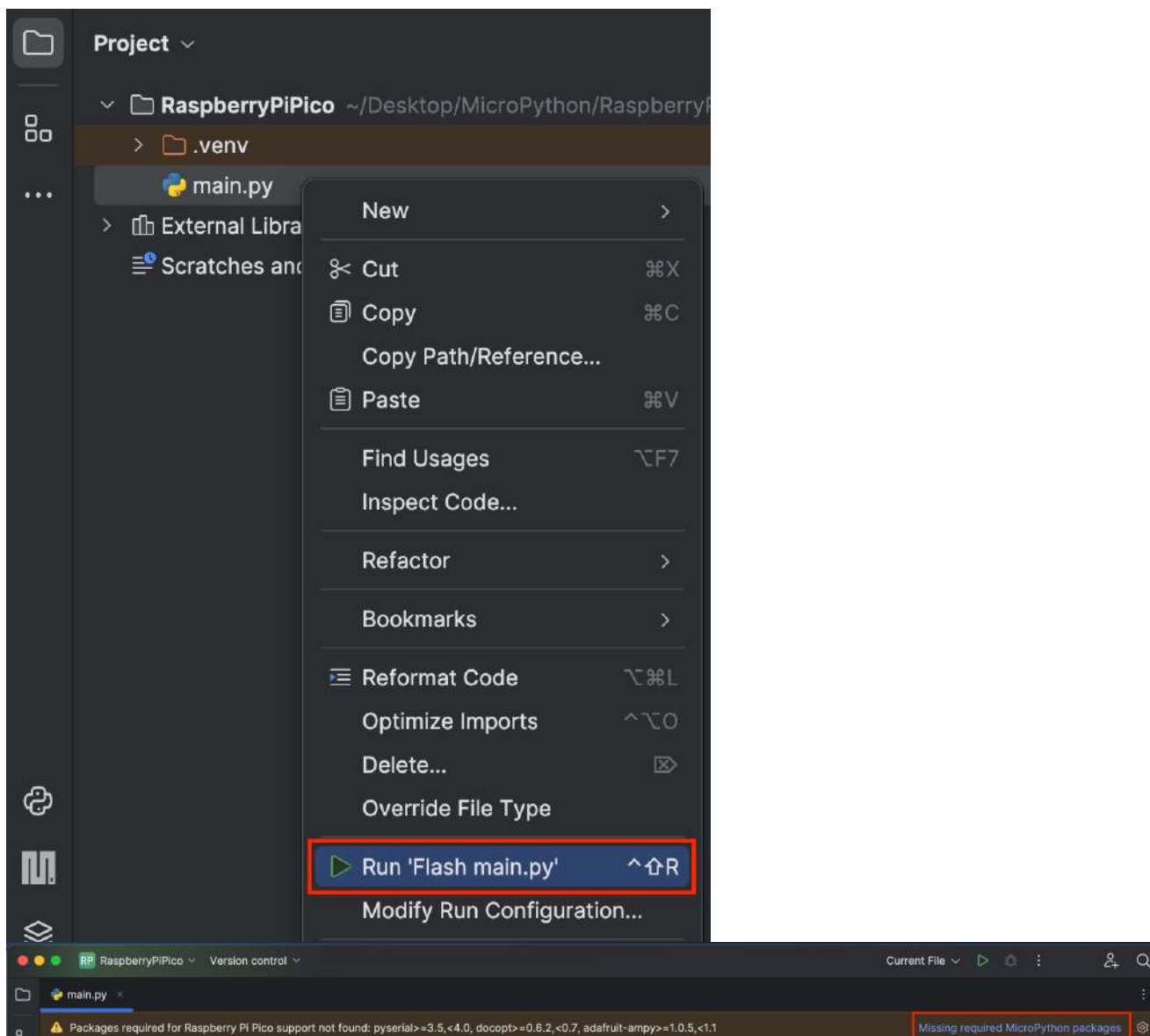


После создания файла PyCharm может предложить установить необходимые дополнительные модули автоматически, для чего необходимо нажать выделенную кнопку (может не сработать в условиях Горного университета):

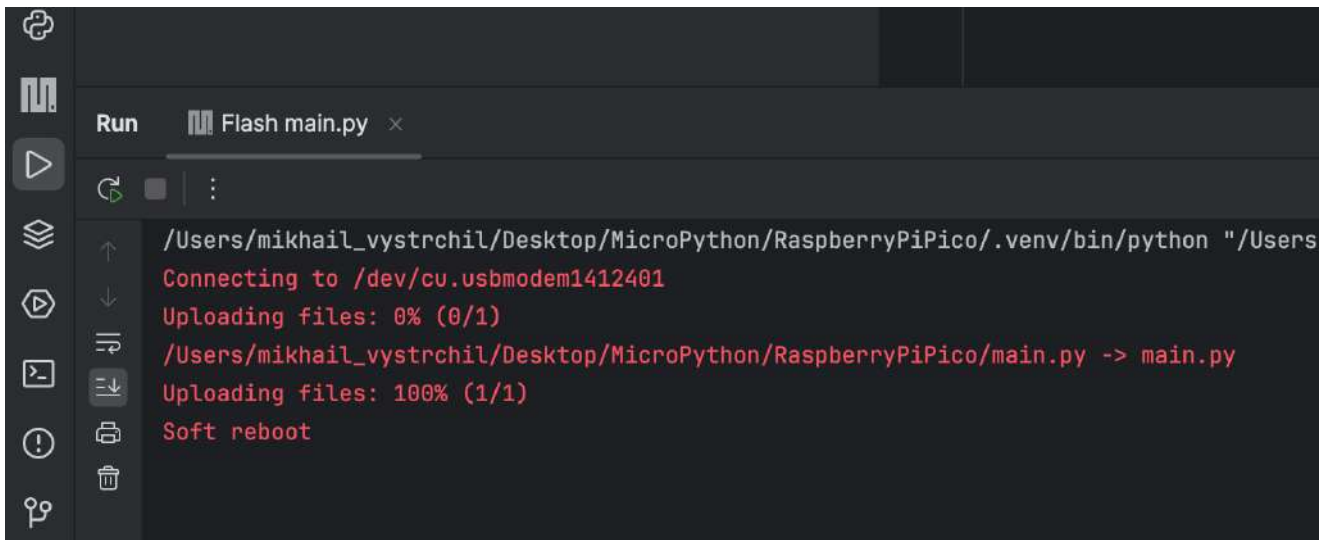


❗ В условиях "Горного университета" не сработает и будет необходимо скачать и установить пакеты вручную!

Проверить, что все прошло успешно, можно, щелкнув ПКМ по созданному файлу и выбрав "Run 'Flash main.py'" (команду, что загружает код в память микроконтроллера):



После чего в консоли должен появиться подобный текст:



Исправление возможных проблем

Если, по какой-то причине, автоматический поиск микроконтроллера не сработал - необходимо прописать его самостоятельно.

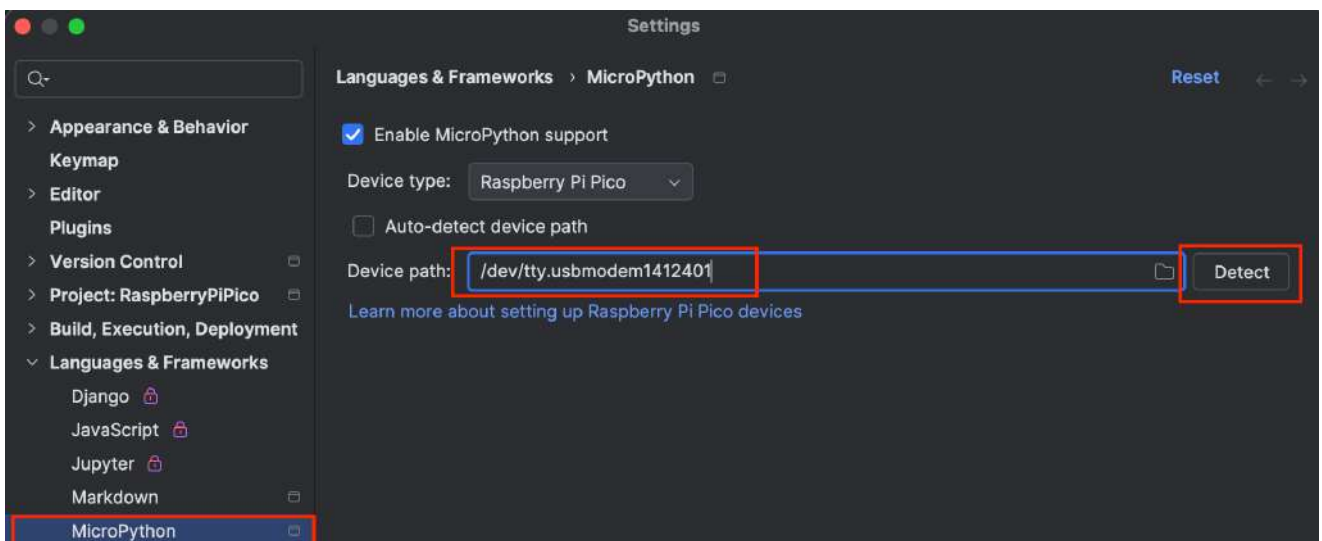
Для Linux и MacOS

Для этого на UNIX системах в терминале нужно "листануть" (1) подключенные порты, введя команду:

```
ls /dev/tty.*
```



В выведенном списке необходимо выбрать наиболее вероятный адрес и скопировать его в "Путь устройства" после чего нажать кнопку "Определить":



Для Windows

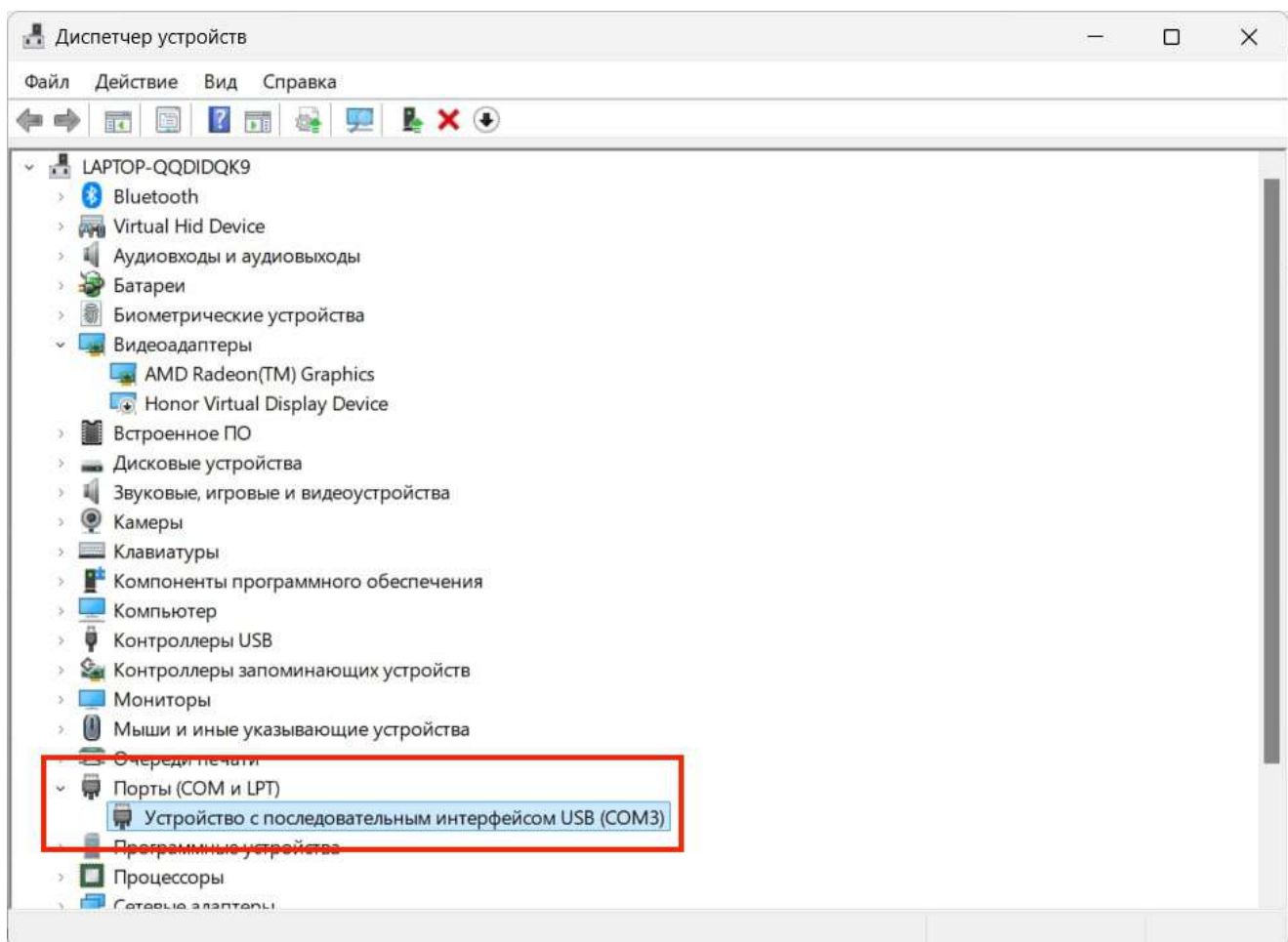
Найти подключенные внешние устройства в ОС Windows можно в "Диспетчере устройств"

Для того чтобы попасть в него следует:

1. Открыть меню "Выполнить" нажав комбинацию клавиш Win + R
2. Ввести команду:

```
devmgmt.msc
```

В выделенной вкладке указывается к какому конкретно порту подключено устройство:



В представленном примере оно подключено к порту COM3.

Первая программа

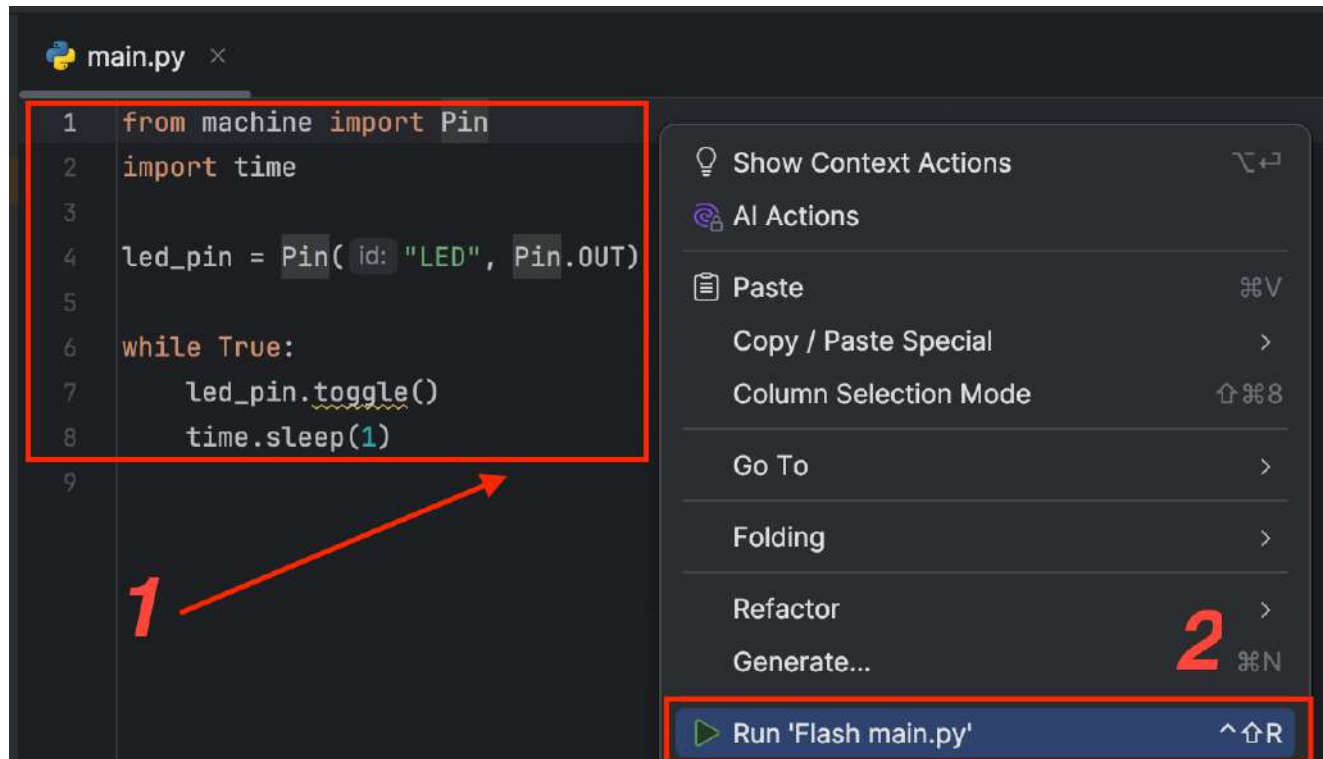
В файле main.py введите следующий код (1) (как он работает и что значит - разберемся далее):

```
from machine import Pin
import time
```

```
led_pin = Pin("LED", Pin.OUT)

while True:
    led_pin.toggle()
    time.sleep(1)
```

После чего в окне щелкните ПКМ и выберете запуск загрузки (2):



В результате встроенный светодиод на микроконтроллере начнет мигать с задержкой в 1 секунду. Если ничего после загрузки кода не произошло, перезагрузите микроконтроллер отключив и подключив его заново.

Поздравляю, вы запрограммировали свое первое устройство!