

Лабораторная работа 2.19

Тема: Работа с файловой системой в Python3 с использованием модуля pathlib.

Цель работы: приобретение навыков по работе с файловой системой с помощью библиотеки pathlib языка программирования Python версии 3.x.

Порядок выполнения работы

1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 RomanGorchakov ▾

Repository name *

/ Test

✓ Test is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

Description (optional)

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

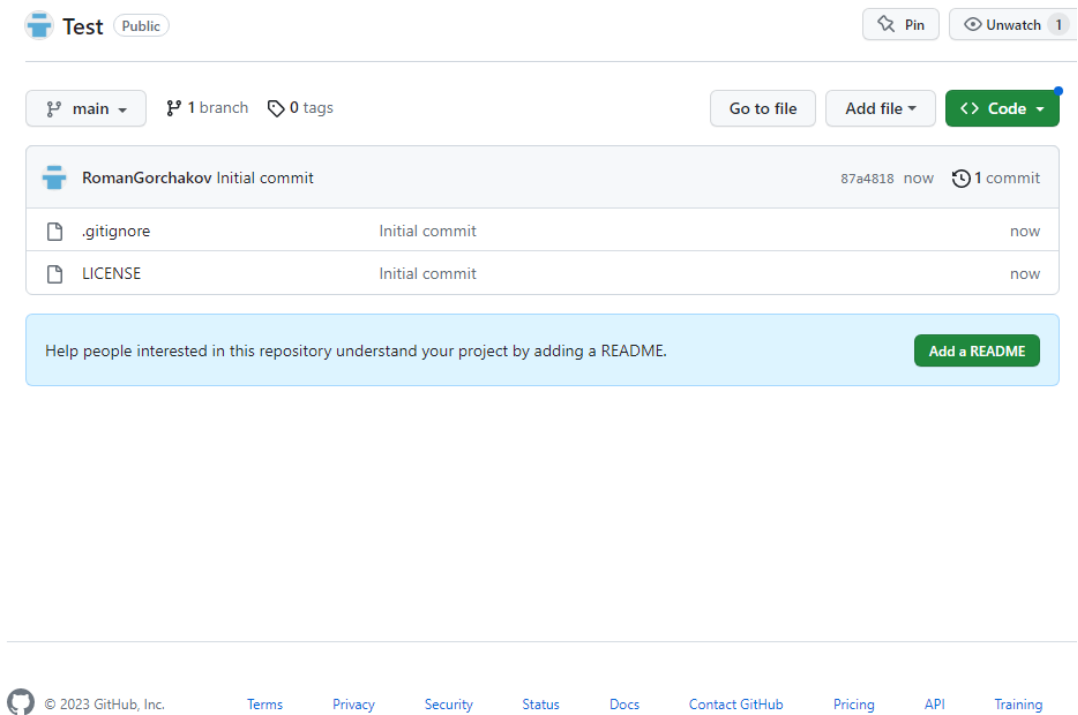
Choose a license

License: MIT License ▾


















A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ You are creating a public repository in your personal account.

Create repository



2. Теперь необходимо дополнить файл `.gitignore` с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «<https://github.com/github/gitignore>» и скачиваем оттуда файл «Python.gitignore».

 Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
 PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
 Plone.gitignore	Covered by global vim template	10 years ago
 Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
 Processing.gitignore	Ignore transpiled .java and .class files (#3016)	4 years ago
 PureScript.gitignore	Update PureScript adding .spago (#3278)	3 years ago
 Python.gitignore	Update Python.gitignore	last year
 Qooxdoo.gitignore	Add gitignore for qooxdoo apps	13 years ago
 Qt.gitignore	Remove trailing whitespace	2 years ago
 R.gitignore	Merge pull request #3792 from jl5000/patch-1	2 years ago
 README.md	Merge pull request #3854 from AnilSeervi/patch-1	2 years ago
 ROS.gitignore	Added ignore for files created by <code>catkin_make_isolated</code>	6 years ago
 Racket.gitignore	Update Racket.gitignore	2 years ago
 Rails.gitignore	Ignore Rails .env according recommendations	2 years ago
 Raku.gitignore	Changes the name of Perl 6 to Raku (#3312)	3 years ago
 RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
 Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files (#3197)	4 years ago

```
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28
```

3. Теперь создаём файл «README.md», где вносим информацию о своей группе и ФИО. Сохраняем набранный текст через кнопку «Commit changes».



4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой репозиторий. После этого организуем репозиторий в соответствии с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout –b develop» для создания ветки разработки; «git branch feature_branch» для создания ветки функций; «git branch release/1.0.0» для создания ветки релиза; «git checkout main» и «git branch hotfix» для создания веток hotfix. Создаём файл .pre-commit-config.yaml и environment.yml.

```
● @RomanGorchakov →/workspaces/Py19 (main) $ git clone https://github.com/RomanGorchakov/Py19.git ClonedPy5.git
Cloning into 'ClonedPy5.git'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), 4.81 KiB | 820.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
● @RomanGorchakov →/workspaces/Py19 (main) $ git checkout -b develop
Switched to a new branch 'develop'
● @RomanGorchakov →/workspaces/Py19 (develop) $ git branch feature_branch
● @RomanGorchakov →/workspaces/Py19 (develop) $ git branch release/1.0.0
● @RomanGorchakov →/workspaces/Py19 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● @RomanGorchakov →/workspaces/Py19 (main) $ git branch hotfix
● @RomanGorchakov →/workspaces/Py19 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/Py19 (develop) $
```

```

Collecting cfgv>=2.0.0 (from pre-commit)
  Downloading cfgv-3.4.0-py2.py3-none-any.whl.metadata (8.5 kB)
Collecting identify>=1.0.0 (from pre-commit)
  Downloading identify-2.5.36-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting nodeenv>=0.11.1 (from pre-commit)
  Downloading nodeenv-1.8.0-py2.py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: pyyaml>=5.1 in /home/codespace/.local/lib/python3.10/site-packages (from pre-commit) (6.0.1)
Collecting virtualenv>=20.10.0 (from pre-commit)
  Downloading virtualenv-20.26.2-py3-none-any.whl.metadata (4.4 kB)
Requirement already satisfied: setuptools in /usr/local/python/3.10.13/lib/python3.10/site-packages (from nodeenv>=0.11.1->pre-commit) (68.2.2)
Collecting distlib<1,>=0.3.7 (from virtualenv>=20.10.0->pre-commit)
  Downloading distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: filelock<4,>=3.12.2 in /home/codespace/.local/lib/python3.10/site-packages (from virtualenv>=20.10.0->pre-commit) (3.13.3)
Requirement already satisfied: platformdirs<5,>=3.9.1 in /home/codespace/.local/lib/python3.10/site-packages (from virtualenv>=20.10.0->pre-commit) (4.2.0)
Downloading pre_commit-3.7.1-py2.py3-none-any.whl (204 kB)
  204.3/204.3 kB 4.6 MB/s eta 0:00:00
Downloading cfgv-3.4.0-py2.py3-none-any.whl (7.2 kB)
Downloading identify-2.5.36-py2.py3-none-any.whl (98 kB)
  99.0/99.0 kB 2.5 MB/s eta 0:00:00
Downloading nodeenv-1.8.0-py2.py3-none-any.whl (22 kB)
Downloading virtualenv-20.26.2-py3-none-any.whl (3.9 MB)
  3.9/3.9 MB 45.8 MB/s eta 0:00:00
Downloading distlib-0.3.8-py2.py3-none-any.whl (468 kB)
  468.9/468.9 kB 10.7 MB/s eta 0:00:00
Installing collected packages: distlib, virtualenv, nodeenv, identify, cfgv, pre-commit
Successfully installed cfgv-3.4.0 distlib-0.3.8 identify-2.5.36 nodeenv-1.8.0 pre-commit-3.7.1 virtualenv-20.26.2
● @RomanGorchakov → /workspaces/Py19 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
● @RomanGorchakov → /workspaces/Py19 (develop) $ conda env export > environment.yml
○ @RomanGorchakov → /workspaces/Py19 (develop) $ 

```

5. Создаём файл «example.py», в котором нужно в котором нужно вывести дерево каталогов.

```

+ show_plane.cpython-312.pyc
+ show_plane.cpython-38.pyc
+ show_plane.cpython-38.pyc.1029953550496
+ show_plane.cpython-38.pyc.1061463643360
+ show_plane.cpython-38.pyc.277182544032
+ show_plane.cpython-38.pyc.396498106528
+ show_plane.cpython-38.pyc.441067754720
+ show_plane.cpython-38.pyc.46053915504
+ show_plane.cpython-38.pyc.642382755520
+ show_plane.cpython-38.pyc.727850612656
+ show_plane.cpython-39.pyc
+ display_plane.py
+ get_plane.py
+ load_plane.py
+ packet.py
+ save_plane.py
+ show_plane.py
+ show_plane.py.bak
+ packet_hard
+ __init__.py
+ __pycache__
+ __init__.cpython-311.pyc
+ __init__.cpython-312.pyc
+ __init__.cpython-38.pyc
+ __init__.cpython-38.pyc.497562744992
+ __init__.cpython-38.pyc.727850612896
+ __init__.cpython-39.pyc
+ display_plane.cpython-38.pyc
+ display_plane.cpython-38.pyc.64238142187
+ display_plane.cpython-38.pyc.72784927876
+ get_plane.cpython-38.pyc
+ get_plane.cpython-38.pyc.642382756000
+ get_plane.cpython-38.pyc.727850612416
+ show_plane.cpython-311.pyc
+ show_plane.cpython-312.pyc
+ show_plane.cpython-38.pyc
+ show_plane.cpython-38.pyc.1029953550496
+ show_plane.cpython-38.pyc.1061463643360
+ show_plane.cpython-38.pyc.277182544032
+ show_plane.cpython-38.pyc.396498106528
+ show_plane.cpython-38.pyc.441067754720
+ show_plane.cpython-38.pyc.46053915504
+ show_plane.cpython-38.pyc.642382755520
+ show_plane.cpython-38.pyc.727850612656
+ show_plane.cpython-39.pyc
+ display_plane.py
+ get_plane.py
+ load_plane.py
+ packet.py
+ save_plane.py
+ show_plane.py
+ show_plane.py.bak
+ schedule-hard.json
+ schedule.json
+ test.md
+ Лабораторная работа 2.19 (22).pdf
+ LP2.19_ГорчаковРВ.docx

```

6. Создаём файл «individual1.py», в котором нужно добавить возможность хранения файла данных в домашнем каталоге пользователя. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по возрастанию номера рейса; вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение. Необходимо также проследить за тем, чтобы файлы генерируемый этой программой не попадали в репозиторий лабораторной работы.

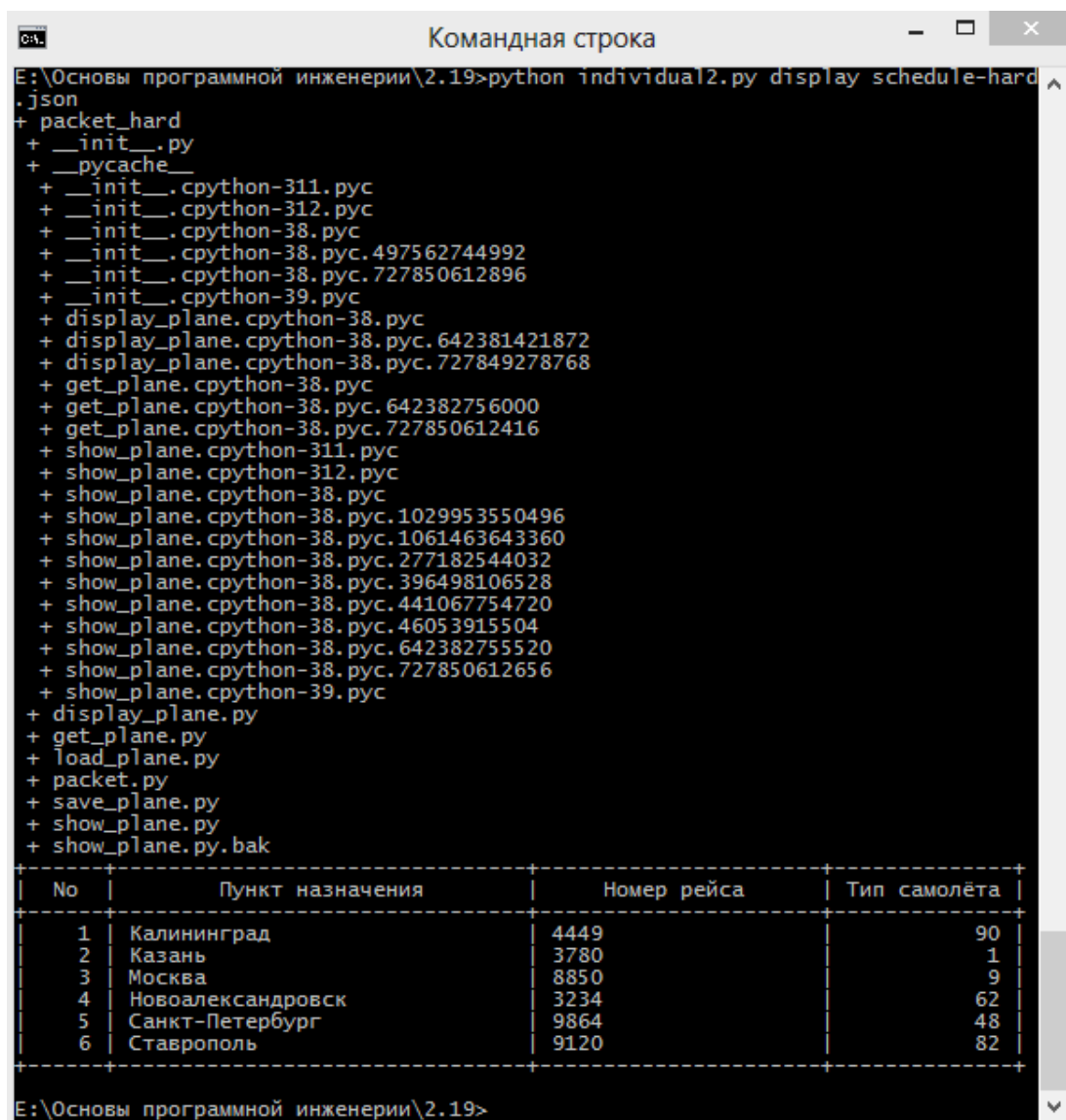
```
Командная строка
E:\Основы программной инженерии\2.19>python individual1.py add schedule.json --race="Калининград" --number=4449 --type=90
E:\Основы программной инженерии\2.19>python individual1.py add schedule.json --race="Казань" --number=3780 --type=1
E:\Основы программной инженерии\2.19>python individual1.py add schedule.json --race="Москва" --number=8850 --type=9
E:\Основы программной инженерии\2.19>python individual1.py add schedule.json --race="Новоалександровск" --number=3234 --type=62
E:\Основы программной инженерии\2.19>python individual1.py add schedule.json --race="Санкт-Петербург" --number=9864 --type=48
E:\Основы программной инженерии\2.19>python individual1.py add schedule.json --race="Ставрополь" --number=9120 --type=82
E:\Основы программной инженерии\2.19>python individual1.py display schedule.json
```

No	Пункт назначения	Номер рейса	Тип самолёта
1	Калининград	4449	90
2	Казань	3780	1
3	Москва	8850	9
4	Новоалександровск	3234	62
5	Санкт-Петербург	9864	48
6	Ставрополь	9120	82

```
E:\Основы программной инженерии\2.19>
```

```
[
  {
    "race": "Калининград",
    "number": 4449,
    "type": 90
  },
  {
    "race": "Казань",
    "number": 3780,
    "type": 1
  },
  {
    "race": "Москва",
    "number": 8850,
    "type": 9
  },
  {
    "race": "Новоалександровск",
    "number": 3234,
    "type": 62
  },
  {
    "race": "Санкт-Петербург",
    "number": 9864,
    "type": 48
  },
  {
    "race": "Ставрополь",
    "number": 9120,
    "type": 82
  }
]
```

7. Создаём файл «individual2.py», в котором нужно разработать аналог утилиты tree в Linux.



```
E:\Основы программной инженерии\2.19>python individual2.py display schedule-hard
.json
+ packet_hard
+ __init__.py
+ __pycache__
+ __init__.cpython-311.pyc
+ __init__.cpython-312.pyc
+ __init__.cpython-38.pyc
+ __init__.cpython-38.pyc.497562744992
+ __init__.cpython-38.pyc.727850612896
+ __init__.cpython-39.pyc
+ display_plane.cpython-38.pyc
+ display_plane.cpython-38.pyc.642381421872
+ display_plane.cpython-38.pyc.727849278768
+ get_plane.cpython-38.pyc
+ get_plane.cpython-38.pyc.642382756000
+ get_plane.cpython-38.pyc.727850612416
+ show_plane.cpython-311.pyc
+ show_plane.cpython-312.pyc
+ show_plane.cpython-38.pyc
+ show_plane.cpython-38.pyc.1029953550496
+ show_plane.cpython-38.pyc.1061463643360
+ show_plane.cpython-38.pyc.277182544032
+ show_plane.cpython-38.pyc.396498106528
+ show_plane.cpython-38.pyc.441067754720
+ show_plane.cpython-38.pyc.46053915504
+ show_plane.cpython-38.pyc.642382755520
+ show_plane.cpython-38.pyc.727850612656
+ show_plane.cpython-39.pyc
+ display_plane.py
+ get_plane.py
+ load_plane.py
+ packet.py
+ save_plane.py
+ show_plane.py
+ show_plane.py.bak
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолёта |
+-----+-----+-----+-----+
| 1 | Калининград | 4449 | 90 |
| 2 | Казань | 3780 | 1 |
| 3 | Москва | 8850 | 9 |
| 4 | Новоалександровск | 3234 | 62 |
| 5 | Санкт-Петербург | 9864 | 48 |
| 6 | Ставрополь | 9120 | 82 |
+-----+-----+-----+-----+
E:\Основы программной инженерии\2.19>
```

8. Выполняем коммит файлов в репозиторий Git в ветку разработки, сливаем её с веткой main и отправляем изменения на сервер GitHub.


```

• @RomanGorchakovo →/workspaces/Py19 (main) $ git push -u
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 2 threads
Compressing objects: 100% (31/31), done.
Writing objects: 100% (32/32), 482.72 KiB | 17.88 MiB/s, done.
Total 32 (delta 10), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (10/10), done.
To https://github.com/RomanGorchakov/Py19
aec6ddc..44856e2 main -> main
branch 'main' set up to track 'origin/main'.
• @RomanGorchakovo →/workspaces/Py19 (main) $

```

Py19

Public

Pin

Unwatch 1

Fork 0

Star 0

Help us improve GitHub Codespaces

Tell us how to make GitHub Codespaces work better for you with three quick questions.

Give feedback

main

1 Branch

0 Tags

Go to file

Add file

Code

RomanGorchakov

522c2e4 · now

6 Commits

Индивидуальные задания

a

now

Отчёт

a

now

Пример

a

now

.gitignore

Create .gitignore

last week

.pre-commit-config.yaml

Changes

last week

LICENSE

Create LICENSE

last week

README.md

a

now

environment.yml

Pathlib

last week

README

MIT license

Горчаков Роман Владимирович. Вариант 4

Лабораторная работа 2.19. Работа с файловой системой в Python3 с использованием модуля pathlib.

Работа с файлами и взаимодействие с файловой системой важны по многим различным причинам. Простейшие случаи могут включать только чтение или запись файлов, но иногда возникают более сложные задачи. Традиционно Python представлял пути к файлам, используя обычные текстовые строки. При поддержке стандартной библиотеки `os.path` это было достаточно, хотя и немного громоздко (как второй пример во введении шоу). Однако, поскольку `paths` не являются строками, важные функции распространяются по всей стандартной библиотеке, включая такие библиотеки, как `os`, `glob` и `shutil`. До Python 3.4 работа с путями файловой системы осуществлялась либо с помощью методов строк (`path.rsplit('\', maxsplit=1)`), либо с помощью модуля `os.path` (`os.path.isfile(os.path.join(os.path.expanduser('~'), 'realpython.txt'))`).

Модуль `pathlib` был введен в Python 3.4 (PEP 428) для решения этих проблем. Он объединяет необходимые функции в одном месте и делает его доступным через методы и свойства простого в использовании объекта `Path`. Всё, что требуется, — это импорт из `pathlib`. Есть несколько разных способов создания пути

About

No description, website, or topics provided.

Readme

MIT license

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

Python 100.0%

Suggested workflows

Based on your tech stack

Python Package using Anaconda

Create and test a Python package on multiple Python versions using Anaconda for package management.

SLSA Generic generator

Generate SLSA3 provenance for your existing release workflows

Контрольные вопросы

1. Какие существовали средства для работы с файловой системой до Python 3.4?

До Python 3.4 работа с путями файловой системы осуществлялась либо с помощью методов строк, либо с помощью модуля `os.path`.

2. Что регламентирует PEP 428?

PEP 428 регламентирует библиотеку `pathlib`.

3. Как осуществляется создание путей средствами модуля `pathlib`?

Прежде всего, существуют `classmethods` наподобие `.cwd()` (текущий рабочий каталог) и `.home()` (домашний каталог пользователя). Путь также может быть явно создан из его строкового представления.

Третий способ построения пути – соединение частей пути с помощью специального оператора `/`. Оператор прямой косой черты используется независимо от фактического разделителя пути на платформе.

4. Как получить путь дочернего элемента файловой системы с помощью модуля `pathlib`?

С помощью метода `.anchor`.

5. Как получить путь к родительским элементам файловой системы с помощью модуля `pathlib`?

С помощью метода `.parent`.

6. Как выполняются операции с файлами с помощью модуля `pathlib`?

Традиционно для чтения или записи файла в Python использовалась встроенная функция `open()`. Это все еще верно, поскольку функция `open()` может напрямую использовать объекты `Path`. Метод `.resolve()` найдет полный путь.

7. Как можно выделить компоненты пути файловой системы с помощью модуля `pathlib`?

Различные части пути удобно доступны как свойства. Основные примеры включают в себя:

- `.name`: имя файла без какого-либо каталога;

- `.parent`: каталог, содержащий файл, или родительский каталог, если путь является каталогом;

- `.stem`: имя файла без суффикса;
- `.suffix` : расширение файла;
- `.anchor`: часть пути перед каталогами.

8. Как выполнить перемещение и удаление файлов с помощью модуля `pathlib`?

Чтобы переместить файл, нужно использовать `.replace()`. Каталоги и файлы могут быть удалены с помощью `.rmdir()` и `.unlink()` соответственно.

9. Как выполнить подсчет файлов в файловой системе?

Есть несколько разных способов перечислить много файлов. Самым простым является метод `.iterdir()` , который перебирает все файлы в данном каталоге. Более гибкие списки файлов могут быть созданы с помощью методов `.glob()` и `.rglob()` (рекурсивный глоб). Например, `pathlib.Path.cwd().glob('*.*txt')` возвращает все файлы с суффиксом `.txt` в текущем каталоге.

10. Как отобразить дерево каталогов файловой системы?

С помощью функции `tree`.

11. Как создать уникальное имя файла?

Сначала укажите шаблон для имени файла с местом для счетчика. Затем проверьте существование пути к файлу, созданного путем соединения каталога и имени файла (со значением счетчика). Если он уже существует, увеличьте счетчик и попробуйте снова.

12. Каковы отличия в использовании модуля `pathlib` для различных операционных систем?

Когда мы создавали экземпляр `pathlib.Path`, возвращался либо объект `WindowsPath`, либо `PosixPath`. Тип объекта будет зависеть от операционной системы, которую вы используете. Эта функция позволяет довольно легко писать кроссплатформенный код. Можно явно запросить `WindowsPath` или `PosixPath`, но вы будете ограничивать свой код только этой системой без каких-либо преимуществ.