

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО РАБОТЕ №2.19
дисциплины «Основы кроссплатформенного программирования»

Выполнил:

Баканов Артем Вадимович

2 курс, группа ИТС-б-о-22-1,

11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: работа с переменными окружениями в python3

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

Порядок выполнения работы:

1. Создала новый репозиторий и клонировала его на свой компьютер.

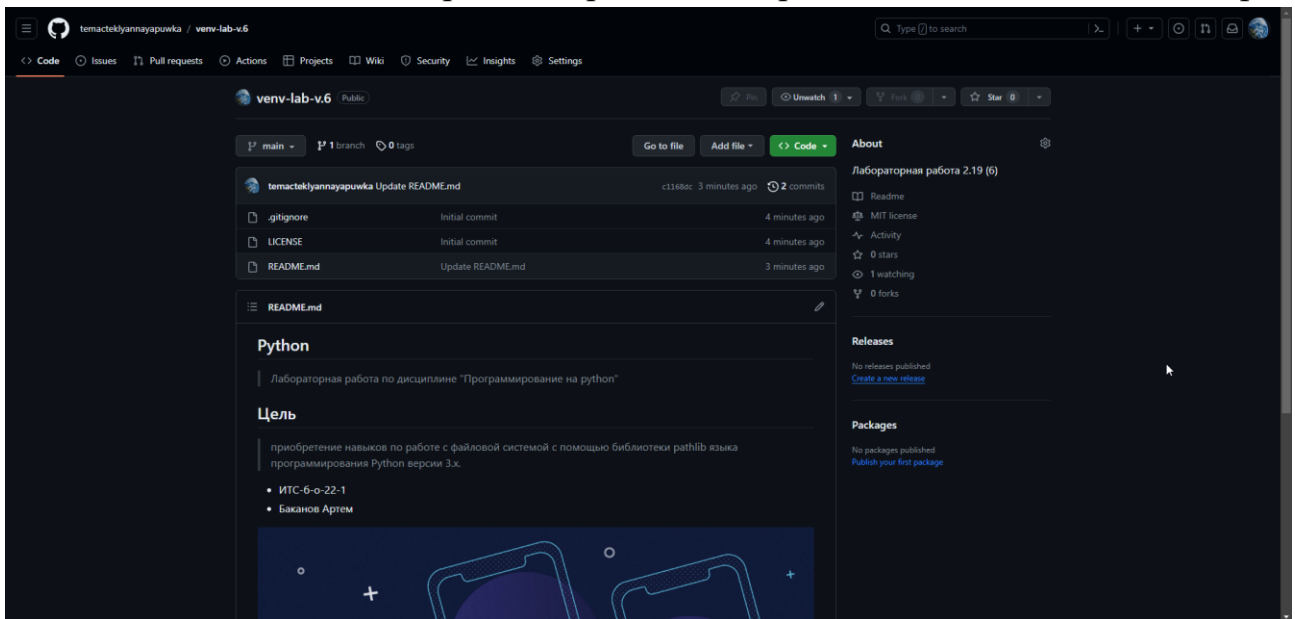


Рисунок – 1. Создан новый репозиторий

2. Клонировал репозиторий на свой компьютер. В ходе данной лабораторной работы работала с моделью ветвления git-flow.

```
Administrator@WIN-CSMACQQ34EQ MINGW64 ~/Desktop
$ git clone https://github.com/temacteklyannayapuwka/venv-lab-v.6.git
Cloning into 'venv-lab-v.6'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
```

Рисунок – 2. Клонирование и модель ветвления git-flow

3. Создал виртуальное окружение Anaconda с именем репозитория.

```
# To deactivate an active environment, use
#
# $ conda deactivate

(base) PS C:\Users\Administrator\Desktop\venv-lab-v.6> conda activate venvlab6
(venvlab6) PS C:\Users\Administrator\Desktop\venv-lab-v.6> conda install isort
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.5.2
  latest version: 23.11.0

Please update conda by running

  $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

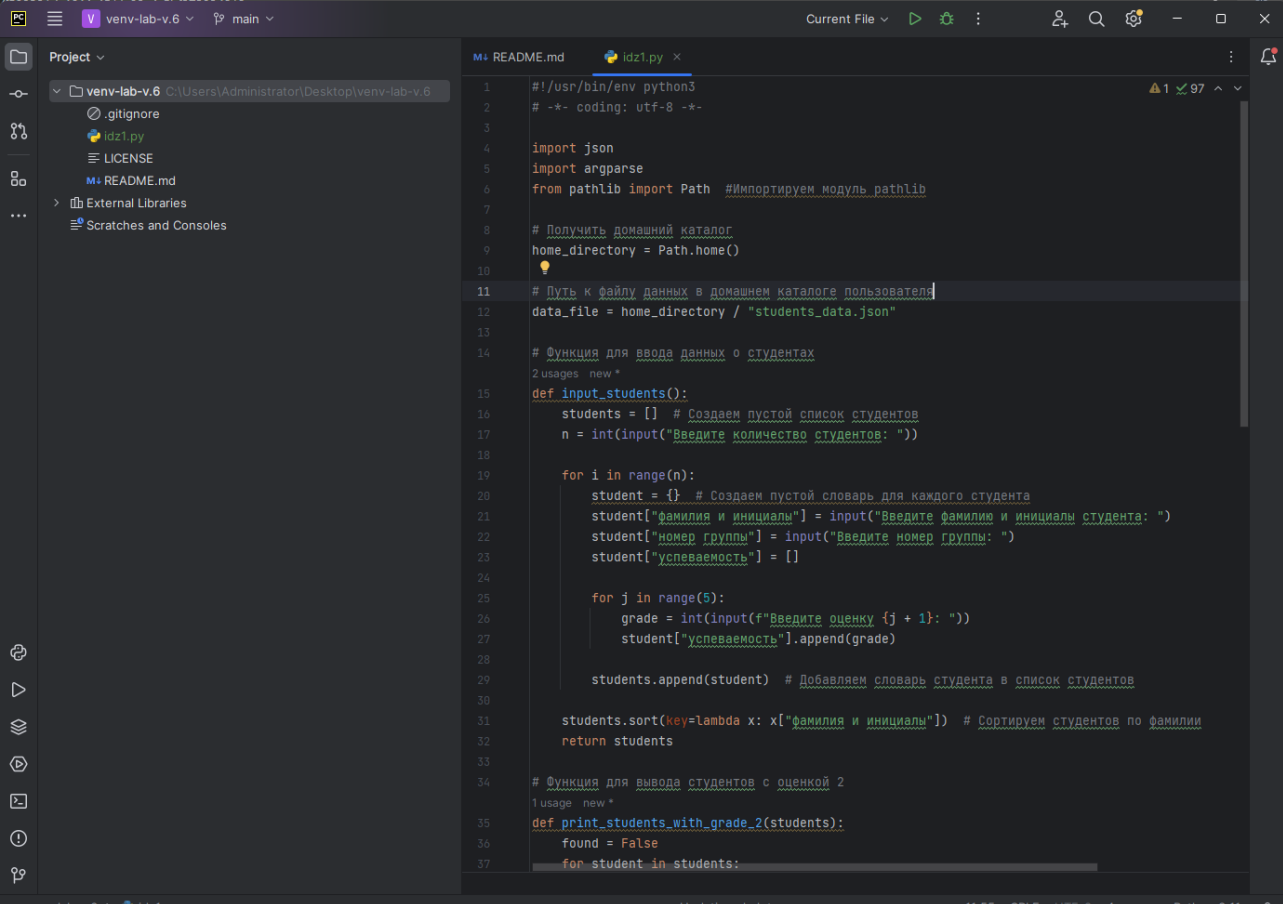
  conda install conda=23.11.0

## Package Plan ##

environment location: C:\Users\Administrator\miniconda3\envs\venvlab6
added / updated specs:
```

Рисунок – 3. Создание виртуального окружения

4. Выполнение индивидуального задания



```
M: README.md  idz1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import json
5  import argparse
6  from pathlib import Path  #Импортируем модуль pathlib
7
8  # Получить домашний каталог
9  home_directory = Path.home()
10
11 # Путь к файлу данных в домашнем каталоге пользователя
12 data_file = home_directory / "students_data.json"
13
14 # Функция для ввода данных о студентах
15 2 usages new *
16 def input_students():
17     students = [] # Создаем пустой список студентов
18     n = int(input("Введите количество студентов: "))
19
20     for i in range(n):
21         student = {} # Создаем пустой словарь для каждого студента
22         student["фамилия и инициалы"] = input("Введите фамилию и инициалы студента: ")
23         student["номер группы"] = input("Введите номер группы: ")
24         student["успеваемость"] = []
25
26         for j in range(5):
27             grade = int(input(f"Введите оценку {j + 1}: "))
28             student["успеваемость"].append(grade)
29
30         students.append(student) # Добавляем словарь студента в список студентов
31
32     students.sort(key=lambda x: x["фамилия и инициалы"]) # Сортируем студентов по фамилии
33     return students
34
35 # Функция для вывода студентов с оценкой 2
36 1 usage new *
37 def print_students_with_grade_2(students):
38     found = False
39     for student in students:
```

Рисунок – 4. Выполнение индивидуального задания

5. Сформировал файлы environment.yml и requirements.txt

```
The following packages will be downloaded:
```

package	build	
flake8-6.1.0	py312haa95532_0	135 KB
pycodestyle-2.11.1	py312haa95532_0	94 KB
pyflakes-3.1.0	py312haa95532_0	168 KB
Total:		397 KB

```
The following NEW packages will be INSTALLED:
```

flake8	pkgs/main/win-64::flake8-6.1.0-py312haa95532_0
mccabe	pkgs/main/noarch::mccabe-0.7.0-pyhd3eb1b0_0
pycodestyle	pkgs/main/win-64::pycodestyle-2.11.1-py312haa95532_0
pyflakes	pkgs/main/win-64::pyflakes-3.1.0-py312haa95532_0

```
Proceed ([y]/n)? y

Downloading and Extracting Packages
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(venvlab6) PS C:\Users\Administrator\Desktop\venv-lab-v.6> conda env export > environment.yml
(venvlab6) PS C:\Users\Administrator\Desktop\venv-lab-v.6> pip freeze > requirements.txt
(venvlab6) PS C:\Users\Administrator\Desktop\venv-lab-v.6>
```

Рисунок – 5. Файлы environment.yml и requirements.txt

6. Отправил на удаленный сервер.

```
new file:   .idea/vcs.xml
new file:   .idea/venv-lab-v.6.iml
new file:   environment.yml
new file:   idz1.py
new file:   requirements.txt

PS C:\Users\Administrator\Desktop\venv-lab-v.6> git commit -m "added new files"
[main a05b48c] added new files
 9 files changed, 115 insertions(+)
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 .idea/venv-lab-v.6.iml
 create mode 100644 environment.yml
 create mode 100644 idz1.py
 create mode 100644 requirements.txt
PS C:\Users\Administrator\Desktop\venv-lab-v.6> git push
Username for 'https://github.com': temacteklyannayapuwka
Password for 'https://temacteklyannayapuwka@github.com':
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (13/13), 3.92 KiB | 3.92 MiB/s, done.
Total 13 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/temacteklyannayapuwka/venv-lab-v.6.git
 c1168dc..a05b48c  main -> main
PS C:\Users\Administrator\Desktop\venv-lab-v.6>
```

Рисунок – 6. Отправка на удаленный сервер

Ссылка: <https://github.com/temacteklyannayapuwka/venv-lab-v.6>

Ответы на контрольные вопросы

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

До Python 3.4 основными средствами для работы с файловой системой были модули:

1. ``os`` - для взаимодействия с операционной системой.
2. ``os.path`` - для операций с путями и файлами.

Эти модули предоставляли функции для работы с директориями, файлами, проверки существования файлов и директорий, а также другие операции с файловой системой.

2. Что регламентирует PEP 428?

PEP 428 регламентирует введение стандартного модуля `pathlib` в Python 3.4. Этот PEP (Python Enhancement Proposal) был предложен Гвидо ван Россумом и осуществлен Армина Ронахера.

Основные цели PEP 428:

Предоставление более удобного и выразительного интерфейса для работы с путями и файловой системой.

Замена более старых и менее удобных способов, таких как использование строковых операций или модуля `os.path`.

Улучшение переносимости кода между различными операционными системами.

`pathlib` вводит классы `Path`, представляющие пути к файлам и директориям, и предоставляет методы и операторы для выполнения различных операций с ними. Внедрение этого модуля значительно улучшило удобство и читаемость кода, связанного с работой с файловой системой.

3. Как осуществляется создание путей средствами модуля pathlib?

Для создания путей с помощью модуля pathlib в Python, вы используете класс Path. Вот несколько примеров создания путей:

```
# Создание объекта Path для файла
```

```
file_path = Path('/путь/к/файлу.txt')
```

```
# Создание объекта Path для директории
```

```
dir_path = Path('/путь/к/директории')
```

```
# Склеивание путей
```

```
new_path = dir_path / 'файл.txt'
```

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

Для получения пути дочернего элемента файловой системы с помощью модуля pathlib вы можете использовать оператор / или метод .joinpath(). Вот примеры обоих способов:

Оператор /:

```
pythonCopy code
```

```
from pathlib import Path # Создание объекта Path для директории dir_path = Path('/путь/к/родительской_директории') # Получение пути дочернего элемента child_path = dir_path / 'дочерняя_директория' / 'файл.txt'
```

Метод .joinpath():

```
pythonCopy code
```

```
from pathlib import Path # Создание объекта Path для директории dir_path = Path('/путь/к/родительской_директории') # Получение пути дочернего элемента child_path = dir_path.joinpath('дочерняя_директория', 'файл.txt')
```

Оба способа создают объект Path, представляющий путь к дочернему элементу файловой системы.

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

Для получения пути к родительским элементам файловой системы с помощью модуля pathlib используется атрибут `.parent`. Вот пример:

pythonCopy code

```
from pathlib import Path # Создание объекта Path для файла или директории
file_path = Path('/путь/к/файлу_или_директории') # Получение пути к
родительской директории parent_path = file_path.parent
```

В данном примере, `parent_path` станет объектом `Path`, представляющим родительскую директорию файла или директории, указанных в `file_path`.

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

Модуль `pathlib` предоставляет удобные методы для выполнения операций с файлами. Вот примеры некоторых операций:

Чтение содержимого файла:

```
from pathlib import Path file_path = Path('/путь/к/файлу.txt') # Чтение
содержимого файла content = file_path.read_text()
```

Запись в файл:

```
from pathlib import Path file_path = Path('/путь/к/файлу.txt') # Запись в файл
text_to_write = 'Пример текста для записи в файл.'
file_path.write_text(text_to_write)
```

Добавление текста в конец файла:

```
from pathlib import Path file_path = Path('/путь/к/файлу.txt') # Добавление
текста в конец файла text_to_append = 'Этот текст будет добавлен в конец файла.'
file_path.write_text(file_path.read_text() + text_to_append)
```

Чтение и запись бинарных данных:

```
from pathlib import Path file_path = Path('/путь/к/файлу.bin') # Чтение
бинарных данных из файла binary_content = file_path.read_bytes() # Запись
```

бинарных данных в файл `new_binary_content = b'Новые бинарные данные.'`
`file_path.write_bytes(new_binary_content)`

Открытие файла в контекстном менеджере:

```
from pathlib import Path file_path = Path('/путь/к/файлу.txt') # Открытие  
файла в контекстном менеджере (автоматическое закрытие файла) with  
file_path.open() as file: content = file.read()
```

Эти методы делают работу с файлами более удобной и читаемой в сравнении с использованием старых методов из модуля `open` и `os`.

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

С помощью модуля `pathlib` можно легко выделять компоненты пути файловой системы. Вот примеры выделения различных компонентов:

Имя файла:

```
from pathlib import Path file_path = Path('/путь/к/директории/файл.txt') #  
Получение имени файла file_name = file_path.name print(f'Имя файла:  
{file_name}')
```

Имя директории:

```
from pathlib import Path file_path = Path('/путь/к/директории/файл.txt') #  
Получение имени директории dir_name = file_path.parent.name print(f'Имя  
директории: {dir_name}')
```

Расширение файла:

```
from pathlib import Path file_path = Path('/путь/к/директории/файл.txt') #  
Получение расширения файла file_extension = file_path.suffix print(f'Расширение  
файла: {file_extension}')
```

Без расширения (базовое имя):

```
from pathlib import Path file_path = Path('/путь/к/директории/файл.txt') #  
Получение базового имени файла без расширения base_name = file_path.stem  
print(f'Базовое имя файла: {base_name}')
```


Эти методы предоставляют удобные способы получения различных компонентов пути, что делает код более читаемым и легко поддерживаемым.

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

Перемещение файла:

```
from pathlib import Path # Исходный путь файла source_path =  
Path('/путь/к/исходному_файлу.txt') # Путь для перемещения файла  
destination_path = Path('/путь/к/целевой_директории/новое_имя_файла.txt') #  
Перемещение файла source_path.rename(destination_path)
```

Удаление файла:

```
from pathlib import Path # Путь к файлу для удаления file_path =  
Path('/путь/к/удаляемому_файлу.txt') # Удаление файла file_path.unlink()
```

Эти примеры показывают, как с использованием модуля pathlib можно легко перемещать и удалять файлы. Важно отметить, что при перемещении файла метод `rename` также может использоваться для переименования файла, если новое имя указано в целевом пути.

9. Как выполнить подсчет файлов в файловой системе?

Для выполнения подсчета файлов в файловой системе с помощью модуля `pathlib` вы можете использовать методы `rglob` (рекурсивный поиск) или `glob`. Вот пример подсчета файлов в текущей директории и ее поддиректориях:

```
from pathlib import Path # Путь к директории, для которой мы хотим  
подсчитать файлы directory_path = Path('/путь/к/директории') # Рекурсивный  
подсчет файлов file_count = sum(1 for _ in directory_path.rglob('*') if _.is_file())  
print(f'Общее количество файлов в директории: {file_count}')
```

В этом примере `glob('*')` рекурсивно ищет все файлы в текущей директории и ее поддиректориях, а `is_file()` проверяет, является ли каждый найденный путь файлом.

Если вы хотите подсчитать только файлы в текущей директории (без рекурсии), используйте `glob`:

```
from pathlib import Path # Путь к текущей директории
current_directory = Path()
# Подсчет файлов в текущей директории
file_count = sum(1 for _ in current_directory.glob('*') if _.is_file())
print(f"Общее количество файлов в текущей директории: {file_count}")
```

Оба эти примера помогут вам выполнить подсчет файлов в файловой системе с использованием модуля `pathlib`.

11. Как отобразить дерево каталогов файловой системы?

Для отображения дерева каталогов файловой системы с помощью модуля `'pathlib'` можно использовать рекурсивную функцию. Вот пример, который покажет, как это сделать:

```
```python
from pathlib import Path

def display_directory_tree(directory_path, indent=""):
 current_dir = Path(directory_path)

 print(f"{indent}+-- {current_dir.name}")

 # Рекурсивный обход поддиректорий
 for item in current_dir.iterdir():
 if item.is_dir():
 display_directory_tree(item, indent + ' ')
```

```

else:
 print(f"{indent} |-- {item.name}")

Путь к директории, для которой мы хотим отобразить дерево
root_directory = Path('/путь/к/директории')

Вызываем функцию для отображения дерева
display_directory_tree(root_directory)
...

```

Этот код создает функцию `display\_directory\_tree`, которая рекурсивно обходит директории, начиная с указанной. Для каждого элемента она выводит его имя, а для поддиректорий вызывает сама себя.

Замените `/путь/к/директории` на путь к той директории, для которой вы хотите отобразить дерево.

## 12. Как создать уникальное имя файла?

Для создания уникального имени файла вам часто приходится добавлять к основному имени какие-то уникальные метки, такие как текущее время, случайное число или другие параметры. Модуль `pathlib` предоставляет удобные средства для создания уникальных имен файлов. Вот несколько примеров:

Используя текущее время:

```

from pathlib import Path from datetime import datetime # Определение
основного имени файла base_name = "file" # Получение текущей даты и времени
в строковом формате timestamp = datetime.now().strftime("%Y%m%d%H%M%S")
Создание уникального имени файла unique_name =
f"{base_name}_{timestamp}.txt" # Путь к файлу file_path =
Path('/путь/к/директории') / unique_name

```

Используя модуль uuid (универсальный уникальный идентификатор):

```
from pathlib import Path import uuid # Определение основного имени файла
base_name = "file" # Генерация уникального идентификатора unique_id =
str(uuid.uuid4()) # Создание уникального имени файла unique_name =
f"{base_name}_{unique_id}.txt" # Путь к файлу file_path =
Path('/путь/к/директории') / unique_name
```

Выбор метода зависит от ваших конкретных требований и предпочтений. Оба этих примера создадут уникальные имена файлов, которые могут быть использованы для создания файлов в файловой системе.

### 13. Каковы отличия в использовании модуля pathlib для различных операционных систем?

Модуль pathlib в Python создан с целью обеспечения переносимости кода между различными операционными системами. В основном, отличия в использовании pathlib для различных ОС сводятся к различиям в символах разделителей пути (/ или \), которые используются в путях файловой системы.

**Вывод:** приобрел навыки по работе с файловой системой с помощью библиотеки pathlib языка программирования Python версии 3.x.