

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение  
высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Институт цифрового развития**

**Кафедра информационных систем и технологий**

Отчет по лабораторной работе №9.

Дисциплина: «Основы программной инженерии»

**Выполнил:**

Студент группы ПИЖ-б-о-22-1,

направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Гуртовой Ярослав Дмитриевич

**Проверил:**

Воронкин Р. А.

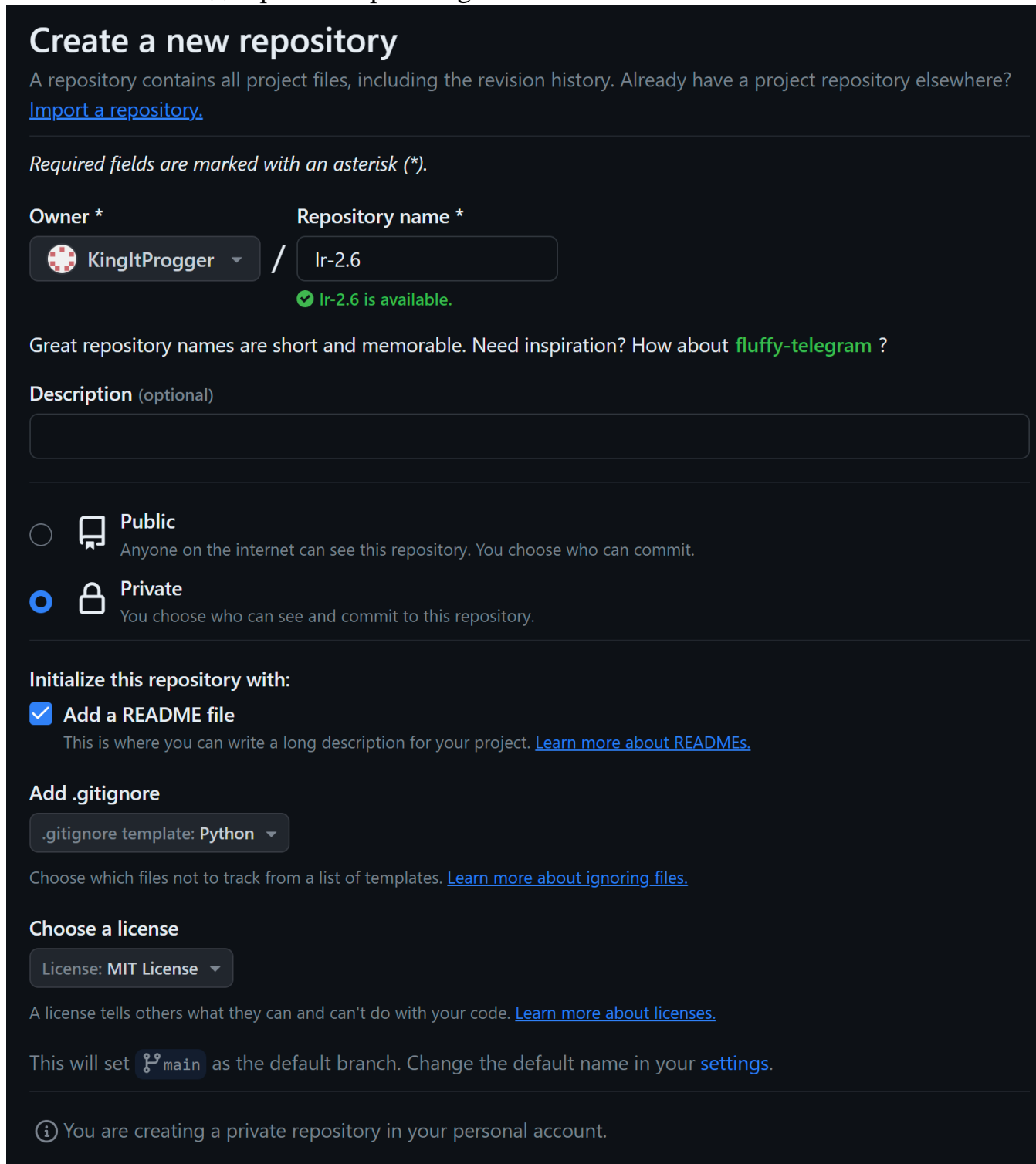
Ставрополь 2023

Тема: Лабораторная работа 2.6 Работа со словарями в языке Python.

Цель работы: приобретение навыков по работе со словарями при написании программ с помощью языка программирования Python версии 3.x.

Выполнение работы:


1. Изучил теоретический материал работы.
2. Создал репозиторий на git.hub.



## 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 \***  KingItProgger / **Repository name \***

✔ lr-2.6 is available.

Great repository names are short and memorable. Need inspiration? How about [fluffy-telegram](#) ?

**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**

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

**Choose a license**

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

This will set  **main** as the default branch. Change the default name in your [settings](#).


 You are creating a private repository in your personal account.

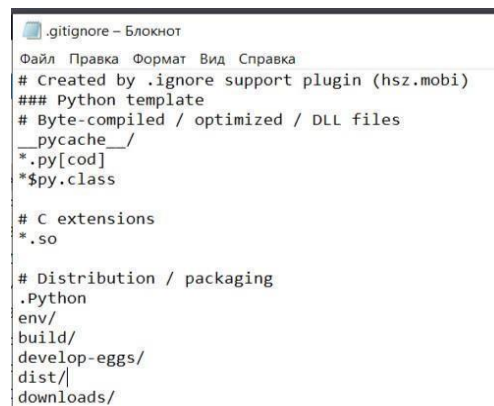
Рисунок 1 – создание репозитория

### 3. Клонировал репозиторий.

```
PS C:\Users\User\Desktop\учеба\3семак\змії\lr_2.6> git clone ht
Cloning into 'lr-2.6'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
PS C:\Users\User\Desktop\учеба\3семак\змії\lr_2.6>
```

Рисунок 2 – клонирование репозитория 4.

Дополнить файл gitignore необходимыми правилами.



```
.gitignore - Блокнот
Файл Правка Формат Вид Справка
# Created by .ignore support plugin (hsz.mobi)
### Python template
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
```

Рисунок 3 – .gitignore для IDE PyCharm

### 5. Организовать свой репозиторий в соответствии с моделью ветвления git-flow.

```
PS C:\Users\User\Desktop\учеба\3семак\змії\lr_2.6\lr-2.6> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Users\User\Desktop\учеба\3семак\змії\lr_2.6\lr-2.6>
```

Рисунок 4 – создание ветки develop

### 6. Проработал примеры из методички.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      # Список работников.
8      workers = []
9
10     # Организовать бесконечный цикл запроса команд.
11     while True:
12         # Запросить команду из терминала.
13         command = input(">>> ").lower()
14         # Выполнить действие в соответствии с командой.
15
16         if command == 'exit':
17             break
18
19         elif command == 'add':
20             # Запросить данные о работнике.
21             name = input("Фамилия и инициалы? ")
22             post = input("Должность? ")
23             year = int(input("Год поступления? "))
24
25             # Создать словарь.
26             worker = {'name': name, 'post': post, 'year': year}
27
28             # Добавить словарь в список.
29             workers.append(worker)
30
31             # Отсортировать список в случае необходимости.
32             if len(workers) > 1:
33                 workers.sort(key=lambda item: item.get('name', ''))
34
35         elif command == 'list':
36             # Заголовок таблицы.
37             line = '+--{}-+--{}-+--{}-+--{}-+'.format(
38                 *args: '-' * 4,
39                 '-' * 30,
40                 '-' * 20,
41                 '-' * 8
42             )
43             print(line)
44
45             # Вывести данные о всех сотрудниках.
46             for idx, worker in enumerate(workers, 1):
47                 print(
48                     '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
49                         *args: idx,
50                         worker.get('name', ''),
51                         worker.get('post', ''),
52                         worker.get('year', 0)
53                     )
54                 )
55                 print(line)
56
57         elif command.startswith('select '):

```

```

58
59     # Получить текущую дату.
60     today = date.today()
61
62     # Разбить команду на части для выделения номера года.
63     parts = command.split( sep: ' ', maxsplit=1)
64
65     # Получить требуемый стаж.
66     period = int(parts[1])
67
68     # Инициализировать счетчик.
69     count = 0
70
71     # Проверить сведения работников из списка.
72     for worker in workers:
73         if today.year - worker.get('year', today.year) >= period:
74             count += 1
75     print(
76         '{:>4}: {}'.format( *args: count, worker.get('name', ''))
77     )
78
79     # Если счетчик равен 0, то работники не найдены.
80     if count == 0:
81         print("Работники с заданным стажем не найдены.")
82
83     elif command == 'help':
84         # Вывести справку о работе с программой.
85
86         print("Список команд:\n")
87         print("add - добавить работника;")
88         print("list - вывести список работников;")
89         print("select <стаж> - запросить работников со стажем;")
90         print("help - отобразить справку;")
91         print("exit - завершить работу с программой.")
92
93     else:
94         print(f"Неизвестная команда {command}", file=sys.stderr)

```

Рисунок 5 – пример 1

Рисунок 6 – пример выполнения примера 1

7. Решите задачу: создайте словарь, связав его с переменной school , и наполните данными, которые бы отражали количество учащихся в разных классах (1а, 1б, 2б, 6а, 7в и т. п.). Внесите изменения в словарь согласно следующему: а) в одном из классов изменилось количество учащихся, б) в школе появился новый класс, с) в школе был расформирован (удален) другой класс. Вычислите общее количество учащихся в школе.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      school = {'1a': 25, '1b': 27, '2b': 30, '6a': 24, '7v': 28,}
8      school['1a'] = 30
9      school['3c'] = 26
10     del school['2b']
11
12     # Находим сумму
13     total_students = sum(school.values())
14
15     # Вывод полученных значений
16     print(school)
17     print(f"Общее количество учащихся в школе: {total_students}")
18
```

Рисунок 7 – задание 9

```
C:\Users\User\AppData\Local\Programs\Python\Python3
{'1a': 30, '1b': 27, '6a': 24, '7v': 28, '3c': 26}
Общее количество учащихся в школе: 135

Process finished with exit code 0
```

Рисунок 8 – пример выполнения 9 задания

8. Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод items(), с с помощью полученного объекта dict\_items создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.

```

1  ▾ #!/usr/bin/env python3
2    # -*- coding: utf-8 -*-
3
4    import sys
5
6  ▾ ▢ if __name__ == '__main__':
7      m = {1: "A", 2: "B", 3: "C", 4: "D"}
8      dict_items = {v: k for k, v in m.items()}
9
10     print(dict_items)
11

```

Рисунок 9 – выполнение задания 11

```

C:\Users\User\AppData\Local\Progra
{'A': 1, 'B': 2, 'C': 3, 'D': 4}

Process finished with exit code 0

```

Рисунок 10 – результат выполнения задания 11

9. Использовать словарь, содержащий следующие ключи: название пункта назначения рейса; номер рейса; тип самолета. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения; вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение.

```
9zadanie.py 11zadanie.py individ1.py x primer1.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      print("Список команд:\n")
8      print("add - добавить рейс;")
9      print("list - вывести список рейсов;")
10     print("select <тип> - вывод на экран пунктов назначения и номеров рейсов для данного типа самолёта")
11     print("help - отобразить справку;")
12     print("exit - завершить работу с программой.")
13
14     # Список работников.
15     aircrafts = []
16
17     # Организовать бесконечный цикл запроса команд.
18     while True:
19         # Запросить команду из терминала.
20         command = input(">>> ").lower()
21         # Выполнить действие в соответствие с командой.
22
23         if command == 'exit':
24             break
25
26         elif command == 'add':
27             # Запросить данные о работнике.
28             name = input("Название пункта назначения рейса? ")
29             number = int(input("Номер рейса? "))
30             tip = input("Тип самолёта? ")
31
32             # Создать словарь.
33             i = {'name': name, 'number': number, 'tip': tip}
34
35             # Добавить словарь в список.
36             aircrafts.append(i)
37
38             # Отсортировать список в случае необходимости.
39             if len(aircrafts) > 1:
40                 aircrafts.sort(key=lambda item: item.get('name', ''))
41
42         elif command == 'list':
43             # Заголовок таблицы.
44             line = '+-{}-+-{}-+-{}-+-{}-+'.format(
45                 *args: '-' * 4,
46                 '-' * 30,
47                 '-' * 20,
48                 '-' * 8
49             )
50             print(line)
51
52             # Вывести данные о всех сотрудниках.
53             for idx, i in enumerate(aircrafts, 1):
54                 print(
55                     '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
56                         *args: idx,
57                         i.get('name', ''),
58                         i.get('number', ''),
59                         i.get('tip', '')
60                     )
61                 )
62             print(line)
63
64         elif command.startswith('select '):
65             # Разбить команду на части для выделения номера года.
66             parts = command.split(sep: ' ', maxsplit=1)
67
68             # Проверить сведения работников из списка.
69             count = 0
70             for i in aircrafts:
71                 for k, v in i.items():
72                     if v == parts[1]:
73                         print("Пункт назначения - ", i["name"])
74                         print("Номер рейса - ", i["number"])
75                         count += 1
76
77             # Если счетчик равен 0, то работники не найдены.
78             if count == 0:
79                 print("Рейс с заданным типом самолёта не найден.")
80
81         elif command == 'help':
82             # Вывести справку о работе с программой.
83             print("Список команд:\n")
84             print("add - добавить рейс;")
85             print("list - вывести список рейсов;")
86             print("select <тип> - вывод на экран пунктов назначения и номеров рейсов для данного типа самолёт")
87             print("help - отобразить справку;")
88             print("exit - завершить работу с программой.")
89
90         else:
91             print(f"Неизвестная команда {command}")
92
93 if __name__ == '__main__': while True: elif command.startswith('select...: if count == 0
```



Рисунок 11 – выполнение индивидуального задания

```
C:\Users\User\AppData\Local\Programs\Python\Python311\python.exe "C:/Users/User/Desktop/ОПИ/lr
Список команд:

add - добавить рейс;
list - вывести список рейсов;
select <тип> - вывод на экран пунктов назначения и номеров рейсов для данного типа самолёта
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Название пункта назначения рейса? Moscow
Номер рейса? 1
Тип самолета? 777
>>> add
Название пункта назначения рейса? Dubai
Номер рейса? 2
Тип самолета? 777
>>> list
+-----+-----+-----+-----+
| 1 | Dubai | 2 | 777 |
| 2 | Moscow | 1 | 777 |
+-----+-----+-----+-----+
>>> help
Список команд:

add - добавить рейс;
list - вывести список рейсов;
select <тип> - вывод на экран пунктов назначения и номеров рейсов для данного типа самолёта
help - отобразить справку;
exit - завершить работу с программой.
>>> exit

Process finished with exit code 0
|
```

Рисунок 12 – результат выполнения индивидуального задания

9.Зафиксировал все изменения в github в ветке develop.

```
PS C:\Users\User\Desktop\учеба\Зсемак\эмий\lr_2.6\lr-2.6> git commit -m"laba"
[develop 60c8550] laba
9 files changed, 241 insertions(+)
create mode 100644 pycharm/.idea/.gitignore
create mode 100644 pycharm/.idea/inspectionProfiles/profiles_settings.xml
create mode 100644 pycharm/.idea/lr 2.6.iml
create mode 100644 pycharm/.idea/misc.xml
create mode 100644 pycharm/.idea/modules.xml
create mode 100644 pycharm/ex1.py
create mode 100644 pycharm/my_task.py
create mode 100644 pycharm/task 9.py
create mode 100644 pycharm/task11.py
PS C:\Users\User\Desktop\учеба\Зсемак\эмий\lr_2.6\lr-2.6>
```

10. Слил ветки.

```
PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr_2.6\lr-2.6> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr_2.6\lr-2.6> git merge develop
Updating e6cfe17..60c8550
Fast-forward
 pycharm/.idea/.gitignore          | 3 +
 .../.idea/inspectionProfiles/profiles_settings.xml | 6 ++
 pycharm/.idea/lr 2.6.iml          | 8 ++
 pycharm/.idea/misc.xml            | 4 +
 pycharm/.idea/modules.xml         | 8 ++
 pycharm/ex1.py                    | 93 ++++++
 pycharm/my_task.py                | 92 ++++++
 pycharm/task 9.py                  | 17 ++++
 pycharm/task11.py                  | 10 +++
 9 files changed, 241 insertions(+)
 create mode 100644 pycharm/.idea/.gitignore
 create mode 100644 pycharm/.idea/inspectionProfiles/profiles_settings.xml
 create mode 100644 pycharm/.idea/lr 2.6.iml
 create mode 100644 pycharm/.idea/misc.xml
 create mode 100644 pycharm/.idea/modules.xml
 create mode 100644 pycharm/ex1.py
 create mode 100644 pycharm/my_task.py
 create mode 100644 pycharm/task 9.py
 create mode 100644 pycharm/task11.py
PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr_2.6\lr-2.6>
```

## **Контрольные вопросы:**

### **1. Что такое словари в языке Python?**

Словарь ( dict ) представляет собой структуру данных (которая ещё называется ассоциативный массив), предназначенную для хранения произвольных объектов с доступом по ключу. Данные в словаре хранятся в формате ключ – значение.

### **2. Может ли функция len() быть использована при работе со словарями?**

Да, функция len() может быть использована для работы со словарями в Python. Она возвращает количество элементов (пар ключ-значение) в словаре.

### **3. Какие методы обхода словарей Вам известны?**

- Цикл for по ключам
- Использование метода items(), который возвращает пары ключ-значение
- Обход только ключей с использованием метода keys()
- Обход только значений с использованием метода values()

### **4. Какими способами можно получить значения из словаря по ключу?**

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
for value in my_dict.values():
    print(value)
```

### **5. Какими способами можно установить значение в словаре по ключу?**

```
my_dict = {}
my_dict['ключ'] = 'значение'
```

### **6. Что такое словарь включений?**

Словарь включений аналогичен списковым включениям, за исключением того, что он создаёт объект словаря вместо списка.

```
{x: x * x for x in (1, 2, 3, 4)}
{1: 1, 2: 4, 3: 9, 4: 16}
```

**7. Самостоятельно изучите возможности функции zip() приведите примеры ее использования.**

Функция zip() в Python используется для объединения двух или более итерируемых объектов (списков, кортежей, и т. д.) в один объект, создавая пары значений. Это может быть полезно, когда вам нужно объединить данные из нескольких источников. Вот примеры использования функции zip().

```
names = ['Анна', 'Петр', 'Мария']
scores = [85, 92, 78]
student_data = list(zip(names, scores))
print(student_data)
```

Результат:

```
[('Анна', 85), ('Петр', 92), ('Мария', 78)]
```

**8. Самостоятельно изучите возможности модуля datetime. Каким функционалом по работе с датой и временем обладает этот модуль?**

Модуль datetime в Python предоставляет обширный функционал для работы с датой и временем. Вот некоторые из его основных возможностей

1. Создание объектов даты и времени:

- datetime.date: Представляет дату (год, месяц, день).
- datetime.time: Представляет время (час, минута, секунда, микросекунда).
- datetime.datetime: Представляет комбинацию даты и времени.

2. Получение текущей даты и времени:

- datetime.datetime.now(): Возвращает текущую дату и время.

3. Разбор и форматирование даты и времени:

- datetime.datetime.strptime(): Разбор строки в объект datetime.
- datetime.datetime.strftime(): Преобразование объекта datetime в строку с заданным форматом.

4. Арифметика с датой и временем:

- Можно выполнять операции сложения и вычитания времени и даты, а также вычислять разницу между двумя моментами времени.

5. Работа с таймзонами:

- Модуль datetime поддерживает работу с часовыми поясами и таймзонами.

6. Извлечение информации:

- Можно получать год, месяц, день, часы, минуты, секунды и другую информацию о дате и времени.

7. Выполнение сравнений:

- Можно сравнивать даты и времена на предмет того, какой из них раньше или позже.

8. Работа с интервалами времени:

- Модуль `datetime` поддерживает интервалы времени, которые позволяют выразить продолжительность времени.