МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №2.6 по дисциплине основы программной инженерии

Выполнил:
Кожухов Филипп Денисович,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры
прикладной математики и
компьютерной
безопасности, Воронкин Р.А.

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys
from datetime import date


if __name__ == '__main__':
    # Список работников.
    workers = []
    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()
        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break

        elif command == 'add':
            # Запросить данные о работнике.
            name = input("Фамилия и инициалы? ")
            post = input("Должность? ")
            year = int(input("Год поступления? "))
            # Создать словарь.
            worker = {
                'name': name,
                'post': post,
                'year': year,
            }
            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 8
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                    "No",
                    "Ф.И.О.",
                    "Должность",
                    "Год"
                )
            )
            print(line)

```

```
            # Вывести данные о всех сотрудниках.
            for idx, worker in enumerate(workers, 1):
                print(
                    '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                        idx,
                        worker.get('name', ''),
                        worker.get('post', ''),
                        worker.get('year', 0)
                    )
                )
            print(line)

        elif command.startswith('select '):
            # Получить текущую дату.
            today = date.today()
            # Разбить команду на части для выделения номера года.
            parts = command.split(' ', maxsplit=1)
            # Получить требуемый стаж.
            period = int(parts[1])
            # Инициализировать счетчик.
            count = 0
            # Проверить сведения работников из списка.
            for worker in workers:
                if today.year - worker.get('year', today.year) >= period:
                    count += 1
                    print(
                        '{:>4}: {}'.format(count, worker.get('name', ''))
                    )
            # Если счетчик равен 0, то работники не найдены.
            if count == 0:
                print("Работники с заданным стажем не найдены.")

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить работника;")
            print("list - вывести список работников;")
            print("select <стаж> - запросить работников со стажем;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)

```

Рисунок 9.1 - Код примера 1

```
"C:\Users\CMDR Inferion\AppData\Local\Microsoft\WindowsApps\python3.8.exe" "C:/
>>> add
Фамилия и инициалы? Jenkins L.A
Должность? LEEEROOOY
Год поступления? 2000
>>> add
Фамилия и инициалы? Cadillac G.W
Должность? 1969
Год поступления? 1969
>>> add
Фамилия и инициалы? Iecocca L.M
Должность? Designer
Год поступления? 1983
>>> list
+------+--------------------------------+----------------------+----------+
|  No  |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Cadillac G.W                   | 1969                 |     1969 |
|    2 | Iecocca L.M                    | Designer             |     1983 |
|    3 | Jenkins L.A                    | LEEEROOOY            |     2000 |
+------+--------------------------------+----------------------+----------+
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>> select 40
   1: Cadillac G.W
>>> select 30
   1: Cadillac G.W
   2: Iecocca L.M
>>> exit

Process finished with exit code 0
```

Рисунок 9.2 - Вывод программы

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys


if __name__ == '__main__':
    school = {}
    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break

        elif command == 'add':
            class_name = input("Введите класс ")
            pupils = int(input("Введите кол-во учащихся в данном классе "))
            school[class_name] = pupils

        elif command == 'list':
            line = '+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} |'.format(
                    "No",
                    "Класс",
                    "Количество учеников",
                )
            )
            print(line)
            index = 0
            for class_name, pupils in school.items():
                index += 1
                print(
                    '| {:>4} | {:<30} | {:<20} |'.format(
                        index,
                        class_name,
                        pupils,
                    )
                )
            print(line)

        elif command == 'edit':
            class_name = input("Введите класс, в котором нужно внести "
                               "изменения ")
            pupils = input("Введите новое количество учащихся в классе "
                           f"{class_name} ")
            school[class_name] = pupils
            print("Количество учащихся было успешно изменено!")

        elif command == 'delete':
            class_name = input("Введите класс, который нужно расформировать ")
            del school[class_name]
            print("Класс был успешно расформирован")

        elif command == 'pupils':
            sum_of_pupils = 0
            for class_name in school:
                sum_of_pupils += int(school[class_name])
            print(f"Количество учеников в школе: {sum_of_pupils}")

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить класс;")
            print("list - вывести список классов;")
            print("edit - изменить количество учащихся в заданном классе.")
            print("delete - расформировать класс.")
            print("pupils - вывести общее количество учеников во всех "
                  "классах.")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)
```

Рисунок 9.3 - Код задачи 1

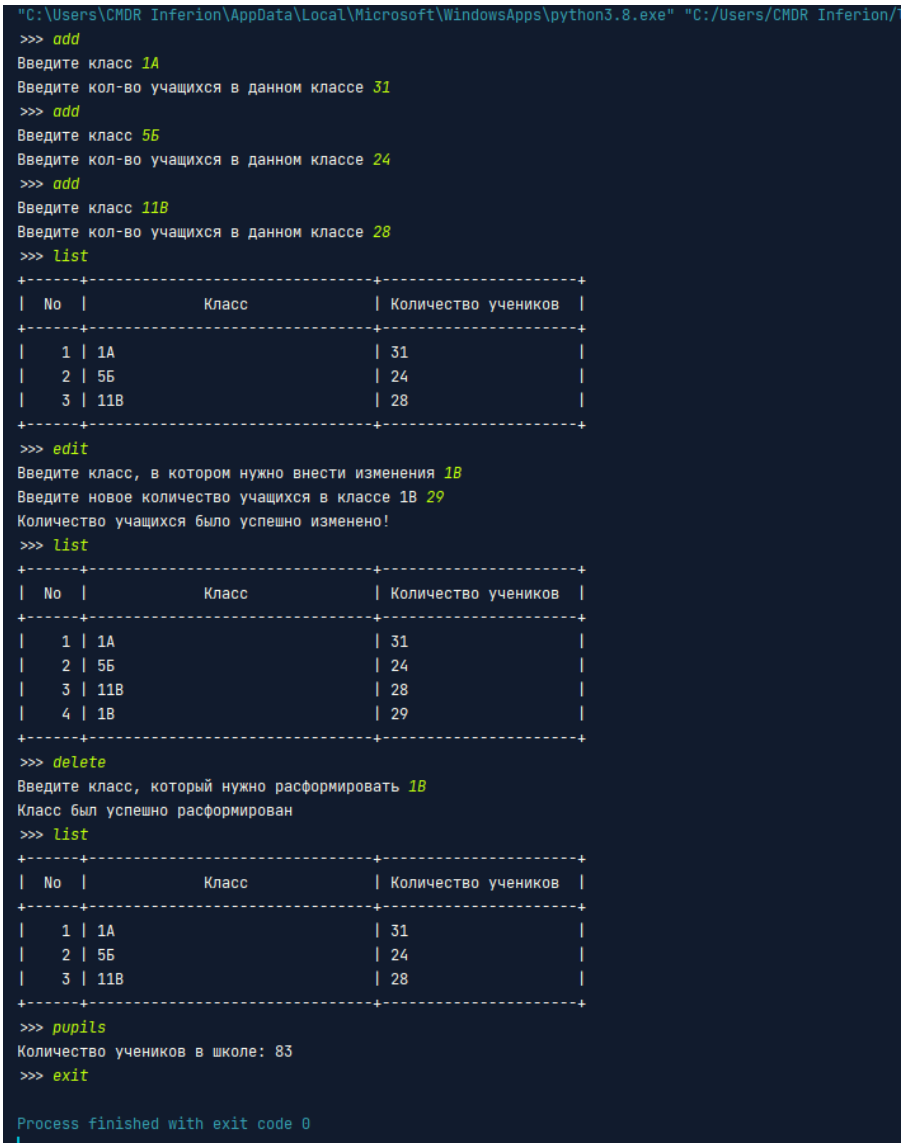


Рисунок 9.4 - Вывод программы

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    dictionary = {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
    print(dictionary)
    swapped = dict(map(reversed, dictionary.items()))
    print(swapped)
```

Рисунок 9.5 - Код задания 2

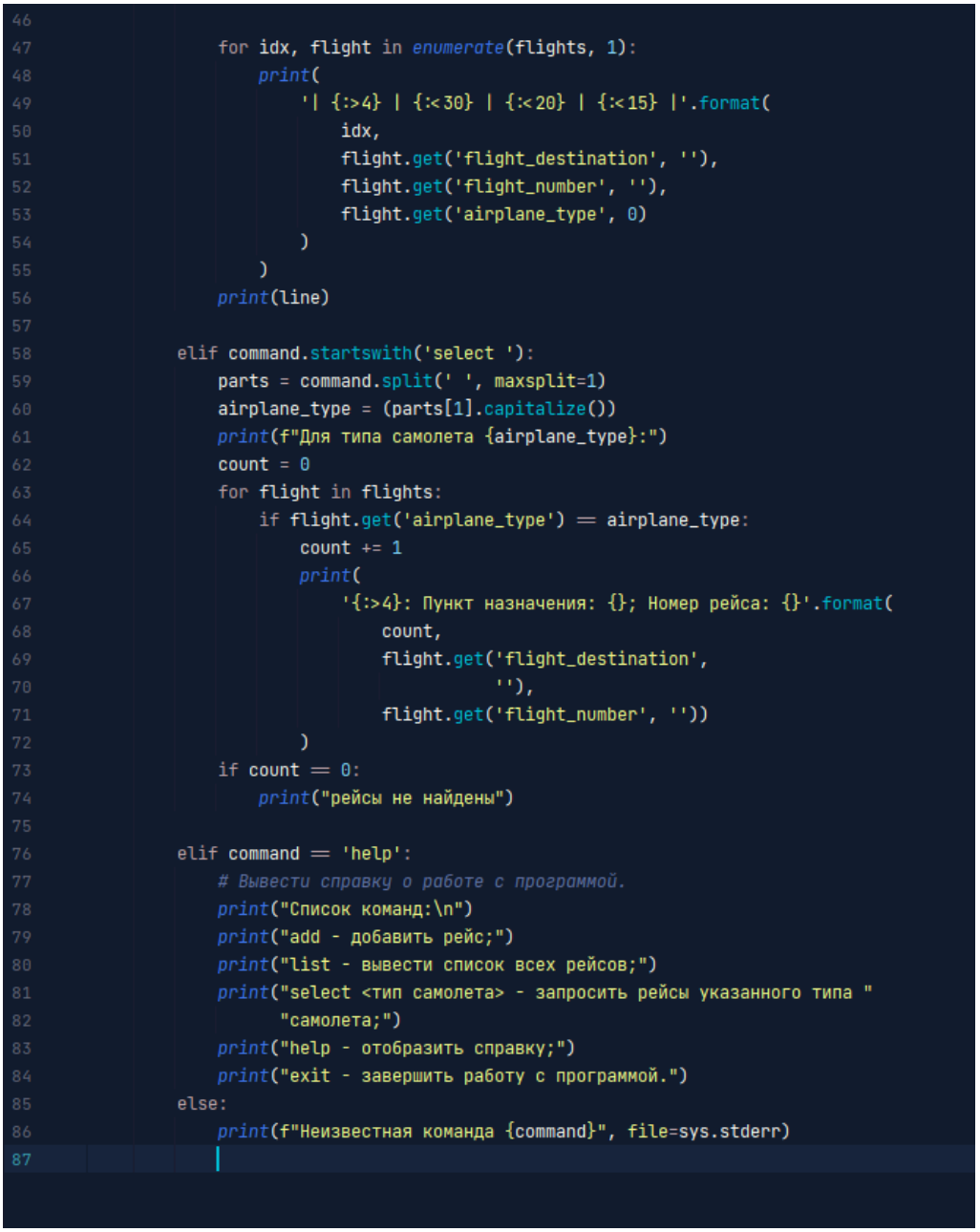
```
"C:\Users\CMDR Inferion\AppData\Local\Microsoft\WindowsApps\python3.
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}

Process finished with exit code 0
```

Рисунок 9.6 - Вывод программы

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys


if __name__ == '__main__':
    flights = []
    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break

        elif command == 'add':
            flight_destination = input("Введите название пункта назначения ")
            flight_number = input("Введите номер рейса ")
            airplane_type = input("Введите тип самолета ")
            flight = {
                'flight_destination': flight_destination,
                'flight_number': flight_number,
                'airplane_type': airplane_type,
            }
            flights.append(flight)
            if len(flights) > 1:
                flights.sort(
                    key=lambda item:
                    item.get('flight_destination', ''))

        elif command == 'list':
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 15
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                    "No",
                    "Пункт назначения",
                    "Номер рейса",
                    "Тип самолета"
                )
            )
            print(line)
```

Рисунок 9.7 - Код ИДЗ (Часть 1)



```
46
47              for idx, flight in enumerate(flights, 1):
48                  print(
49                      '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
50                          idx,
51                          flight.get('flight_destination', ''),
52                          flight.get('flight_number', ''),
53                          flight.get('airplane_type', 0)
54                      )
55                  )
56              print(line)
57
58          elif command.startswith('select '):
59              parts = command.split(' ', maxsplit=1)
60              airplane_type = (parts[1].capitalize())
61              print(f"Для типа самолета {airplane_type}:")
62              count = 0
63              for flight in flights:
64                  if flight.get('airplane_type') == airplane_type:
65                      count += 1
66                      print(
67                          '{:>4}: Пункт назначения: {}; Номер рейса: {}'.format(
68                              count,
69                              flight.get('flight_destination',
70                                         ''),
71                              flight.get('flight_number', ''))
72                      )
73              if count == 0:
74                  print("рейсы не найдены")
75
76          elif command == 'help':
77              # Вывести справку о работе с программой.
78              print("Список команд:\n")
79              print("add - добавить рейс;")
80              print("list - вывести список всех рейсов;")
81              print("select <тип самолета> - запросить рейсы указанного типа "
82                    "самолета;")
83              print("help - отобразить справку;")
84              print("exit - завершить работу с программой.")
85          else:
86              print(f"Неизвестная команда {command}", file=sys.stderr)
87
```

Рисунок 9.8 - Код ИДЗ (Часть 2)

Рисунок 9.9 - Пример вывода программы

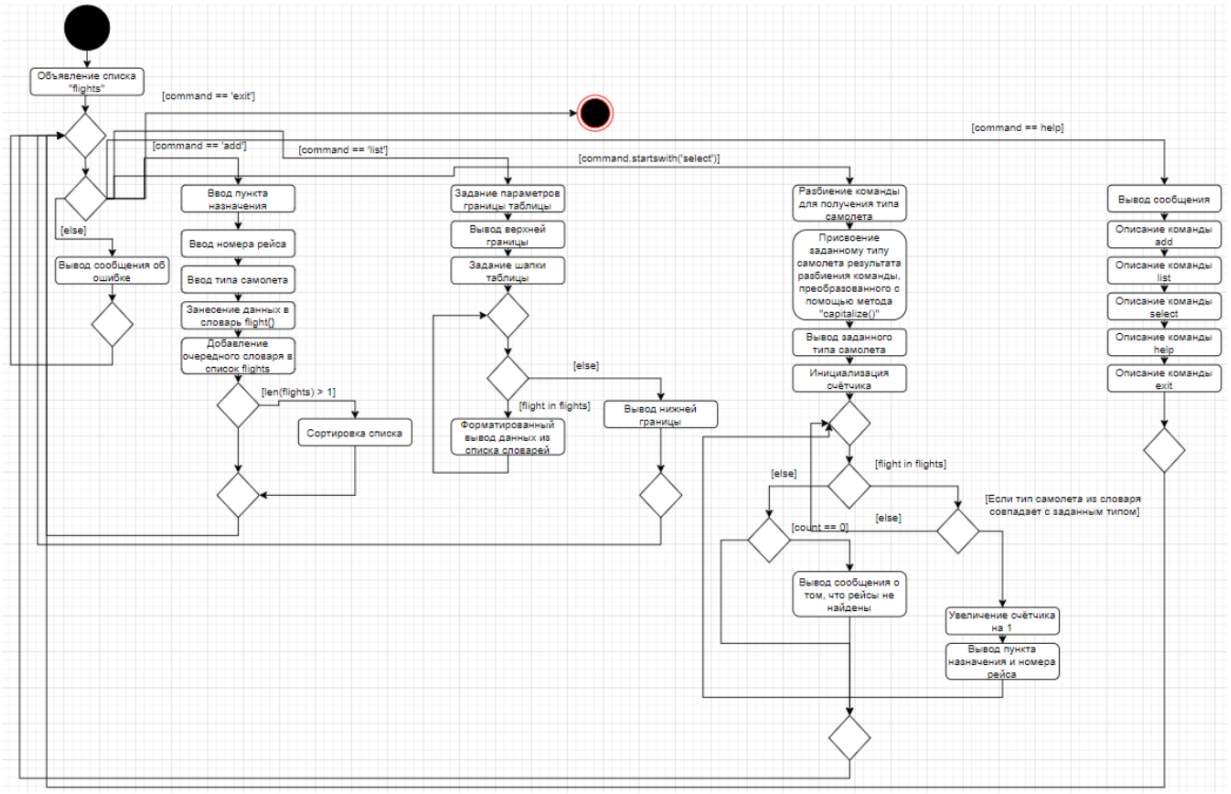


Рисунок 9.10 - UML-диаграмма

ОТВЕТЫ НА ВОПРОСЫ

1. Словарь представляет собой структуру данных (которая ещё называется ассоциативный массив), предназначенную для хранения произвольных объектов с доступом по ключу.

2. Да, len() может быть использован – он выводит количество элементов (пар типа «ключ: элемент»).

3. Перебор ключей в цикле for, перебор элементов в цикле for, одновременный перебор ключей и их значений в цикле for.

4. С помощью метода get(), при обходе в цикле for, используя переменную в качестве счетчика ключей.

5. С помощью метода setdefault(), при непосредственном обращении к ключу словаря.

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

7. Функция zip() в Python создает итератор, который объединяет элементы из нескольких источников данных. Эта функция работает со списками, кортежами, множествами и словарями для создания списков или кортежей, включающих все эти данные. У функции zip() множество сценариев применения. Например, она пригодится, если нужно создать набор словарей из двух массивов, каждый из которых содержит имя и номер сотрудника. Функция zip() принимает итерируемый объект, например, список, кортеж, множество или словарь в качестве аргумента. Затем она генерирует список кортежей, которые содержат элементы из каждого объекта, переданного в функцию. Предположим, что есть список имен и номером сотрудников, и их нужно объединить в массив кортежей. Для этого можно использовать функцию zip().

8. Модуль datetime предоставляет классы для обработки времени и даты разными способами. Поддерживается и стандартный способ представления времени, однако больший упор сделан на простоту манипулирования датой, временем и их частями.
Классы, предоставляемые модулем datetime:

- Класс datetime.date(year, month, day) - стандартная дата.

Атрибуты: year, month, day. Неизменяемый объект.

- Класс datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None) - стандартное время, не зависит от даты.

Атрибуты: hour, minute, second, microsecond, tzinfo.

- Класс datetime.timedelta - разница между двумя моментами времени, с точностью до микросекунд.

- Класс datetime.tzinfo - абстрактный базовый класс для информации о временной зоне (например, для учета часового пояса и / или летнего времени).

- Класс datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None) - комбинация даты и времени.

Обязательные аргументы:
- datetime.MINYEAR (1) ≤ year ≤ datetime.MAXYEAR (9999)
- 1 ≤ month ≤ 12
- 1 ≤ day ≤ количество дней в данном месяце и году

Необязательные:
- 0 ≤ minute < 60
- 0 ≤ second < 60
- 0 ≤ microsecond < 1000000

Методы класса datetime:
- datetime.today() - объект datetime из текущей даты и времени.

Работает также, как и datetime.now() со значением tz=None.
- datetime.fromtimestamp(timestamp) - дата из стандартного представления времени.

- datetime.fromordinal(ordinal) - дата из числа, представляющего собой количество дней, прошедших с 01.01.1970.

- datetime.now(tz=None) - объект datetime из текущей даты и времени.

- datetime.combine(date, time) - объект datetime из комбинации объектов date и time.

- datetime.strptime(date_string, format) - преобразует строку в datetime (так же, как и функция strptime из модуля time).

- datetime.strftime(format) - см. функцию strftime из модуля time.

- datetime.date() - объект даты (с отсечением времени).

- datetime.time() - объект времени (с отсечением даты).

- datetime.replace([year[, month[, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]]]]) - возвращает новый объект datetime с изменёнными атрибутами.

- datetime.timetuple() - возвращает struct_time из datetime.

- datetime.toordinal() - количество дней, прошедших с 01.01.1970.

- datetime.timestamp() - возвращает время в секундах с начала эпохи.

- datetime.weekday() - день недели в виде числа, понедельник - 0, воскресенье - 6.

- datetime.isoweekday() - день недели в виде числа, понедельник - 1, воскресенье - 7.

- datetime.isocalendar() - кортеж (год в формате ISO, ISO номер недели, ISO день недели).

- datetime.isoformat(sep='T') - красивая строка вида "YYYY-MMDDTHH:MM:SS.mmmmmm" или, если microsecond == 0, "YYYY-MM-DDTHH:MM:SS"

- datetime.ctime() - преобразует время, выраженное в секундах с начала эпохи в строку вида "Thu Sep 27 16:42:37 2012".