

**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 №9 по дисциплине основы программной
инженерии**

Выполнила:
Емельянова Яна
Александровна, 2 курс,
группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

1. Работа со словарями в языке Python

Примеры из методических указаний

Создание словарей

```
e1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      a = {'cat': 'кошка', 'dog': 'собака',
6          'bird': 'птица', 'mouse': 'мышь'}
7      print(a)
8      print(a['cat'])
9      print(a['bird'])
10     a['elephant'] = 'бегемот' # добавляем
11     a['table'] = 'стол' # добавляем
12     print(a)
13     a['elephant'] = 'слон' # изменяем
14     del a['table'] # удаляем
15     print(a)
16
```

```
{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь'}
кошка
птица
{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь', 'elephant': 'бегемот', 'table': 'стол'}
{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь', 'elephant': 'слон'}

Process finished with exit code 0
```

Элементы с одинаковыми ключами

```
e1.py x  e2.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      nums = {1: 'one', 2: 'two', 3: 'three'}
6      person = {'name': 'Tom', 1: [30, 15, 16],
7               2: 2.34, ('ab', 100): 'no'}
8      print(nums)
9      print(person)
10
```

```
↓
{1: 'one', 2: 'two', 3: 'three'}
{name: 'Tom', 1: [30, 15, 16], 2: 2.34, ('ab', 100): 'no'}
Process finished with exit code 0
```

Перебор элементов словаря в цикле for

```
e3.py ×
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     nums = {1: 'one', 2: 'two', 3: 'three'}
6     for i in nums:
7         print(i)
8     for i in nums:
9         print(nums[i])
10    n = nums.items()
11    print(n)
12    for key, value in nums.items():
13        print(key, 'is', value)
14    v_nums = []
15    for v in nums.values():
16        v_nums.append(v)
17    print(v_nums)
18    v_nums_2 = [v for v in nums.values()]
19    print(v_nums_2)
20
```

```
1
2
3
one
two
three
dict_items([(1, 'one'), (2, 'two'), (3, 'three')])
1 is one
2 is two
3 is three
['one', 'two', 'three']
['one', 'two', 'three']

Process finished with exit code 0
```

Методы словаря

```
e4.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     nums = {1: 'one', 2: 'two', 3: 'three'}
6     a = {'dog': 'собака', 'cat': 'кошка', 'mouse': 'мышь',
7         'bird': 'птица', 'elephant': 'слон'}
8     print(a)
9     a.clear()
10    print(a)
11    nums2 = nums.copy()
12    nums2[4] = 'four'
13    print(nums)
14    print(nums2)
15    a = [1, 2, 3]
16    c = dict.fromkeys(a)
17    print(c)
18    d = dict.fromkeys(a, 10)
19    print(d)
20    print(nums.get(1))
21    print(nums.pop(1))
22    print(nums)
23    print(nums.popitem())
24    print(nums)
25    print(nums.setdefault(4, 'four'))
26    print(nums)
27    nums.update({6: 'six', 7: 'seven'})
28    print(nums)
29
```

```

{'dog': 'собака', 'cat': 'кошка', 'mouse': 'мышь', 'bird': 'птица', 'elephant': 'слон'}
{}
{1: 'one', 2: 'two', 3: 'three'}
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
{1: None, 2: None, 3: None}
{1: 10, 2: 10, 3: 10}
one
one
{2: 'two', 3: 'three'}
(3, 'three')
{2: 'two'}
four
{2: 'two', 4: 'four'}
{2: 'two', 4: 'four', 6: 'six', 7: 'seven'}

Process finished with exit code 0

```

Словарь включений

```

e5.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     a = {x: x * x for x in (1, 2, 3, 4)}
6     print(a)
7     b = dict((x, x * x) for x in (1, 2, 3, 4))
8     print(b)
9     c = {name: len(name) for name in ('Stack', 'Overflow', 'Exchange') if
10         len(name)
11         > 6}
12     print(c)
13     d = dict((name, len(name)) for name in ('Stack', 'Overflow', 'Exchange') if
14         len(name) > 6)
15     print(d)
16     initial_dict = {'x': 1, 'y': 2}
17     e = {key: value for key, value in initial_dict.items() if key == 'x'}
18     print(e)
19     my_dict = {1: 'a', 2: 'b', 3: 'c'}
20     swapped = dict(map(reversed, my_dict.items()))
21     print(swapped)
22
23

```

```
↓ {1: 1, 2: 4, 3: 9, 4: 16}
⏮ {1: 1, 2: 4, 3: 9, 4: 16}
⏮ {'Overflow': 8, 'Exchange': 8}
⏮ {'Overflow': 8, 'Exchange': 8}
🖨 {'x': 1}
🗑 {'a': 1, 'b': 2, 'c': 3}

Process finished with exit code 0
```

Объединение словарей

```
е6.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     dict1 = {'w': 1, 'x': 1}
6     dict2 = {'x': 2, 'y': 2, 'z': 2}
7     dict3 = {k: v for d in [dict1, dict2] for k, v in d.items()}
8     print(dict3)
9
```

```
↓ {'w': 1, 'x': 2, 'y': 2, 'z': 2}
⏮
⏮
🖨
🗑

Process finished with exit code 0
```

1.1 Пример 1(рис 1, 2, 3, 4).

```
ex1.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5 from datetime import date
6
7 ▶ if __name__ == '__main__':
8     # Список работников.
9     workers = []
10    # Организовать бесконечный цикл запроса команд.
11    while True:
12        # Запросить команду из терминала.
13        command = input(">>> ").lower()
14        # Выполнить действие в соответствие с командой.
15        if command == 'exit':
16            break
17
18        elif command == 'add':
19            # Запросить данные о работнике.
20            name = input("Фамилия и инициалы? ")
21            post = input("Должность? ")
22            year = int(input("Год поступления? "))
23            # Создать словарь.
24            worker = {
25                'name': name,
26                'post': post,
27                'year': year,
28            }
29            # Добавить словарь в список.
30            workers.append(worker)
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                '-' * 4,
39                '-' * 30,
```

Рисунок 1 – Код программы

```

ex1.py x
37 line = '+-{}--{}--{}--{}--'.format(
38     '-' * 4,
39     '-' * 30,
40     '-' * 20,
41     '-' * 8
42 )
43 print(line)
44 print(
45     '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
46         "No",
47         "Ф.И.О.",
48         "Должность",
49         "Год"
50     )
51 )
52 print(line)
53
54 # Вывести данные о всех сотрудниках.
55 for idx, worker in enumerate(workers, 1):
56     print(
57         '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
58             idx,
59             worker.get('name', ''),
60             worker.get('post', ''),
61             worker.get('year', 0)
62         )
63     )
64     print(line)
65
66 elif command.startswith('select '):
67     # Получить текущую дату.
68     today = date.today()
69     # Разбить команду на части для выделения номера года.
70     parts = command.split(' ', maxsplit=1)
71     # Получить требуемый стаж.
72     period = int(parts[1])
73     # Инициализировать счетчик.
74     count = 0
75     # Проверить сведения работников из списка.
76
if __name__ == '__main__': while True

```

Рисунок 2 – Код программы, продолжение


```

75     # Проверить сведения работников из списка.
76     for worker in workers:
77         if today.year - worker.get('year', today.year) >= period:
78             count += 1
79             print(
80                 '{:>4}: {}'.format(count, worker.get('name', ''))
81             )
82     # Если счетчик равен 0, то работники не найдены.
83     if count == 0:
84         print("Работники с заданным стажем не найдены.")
85
86     elif command == 'help':
87         # Вывести справку о работе с программой.
88         print("Список команд:\n")
89         print("add - добавить работника;")
90         print("list - вывести список работников;")
91         print("select <стаж> - запросить работников со стажем;")
92         print("help - отобразить справку;")
93         print("exit - завершить работу с программой.")
94     else:
95         print(f"Неизвестная команда {command}", file=sys.stderr)
96

```

Рисунок 3 – Код программы, продолжение

```
>>> add
Фамилия и инициалы? Ehjkh E.H
Должность? Qpksnk
Год поступления? 1980
>>> add
Фамилия и инициалы? Ertvk A.D
Должность? HJkkn
Год поступления? 2017
>>> add
Фамилия и инициалы? Yutjgk D.F
Должность? Tyhfgk
Год поступления? 1990
>>> add
Фамилия и инициалы? Tgkhhj A.D
Должность? Qpksnk
Год поступления? 2000
>>> list
```

No	Ф.И.О.	Должность	Год
1	Ehjkh E.H	Qpksnk	1980
2	Ertvk A.D	HJkkn	2017
3	Tgkhhj A.D	Qpksnk	2000
4	Yutjgk D.F	Tyhfgk	1990

```
>>> select 10
1: Ehjkh E.H
2: Tgkhhj A.D
3: Yutjgk D.F
>>> jhj
>>> Неизвестная команда jhj
exit

Process finished with exit code 0
```

Рисунок 4 – Работа программы

1.2 Задача 1 (рис. 5, 6, 7, 8, 9, 10, 11, 12).

```

1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 ▶ if __name__ == '__main__':
7     school = {}
8     while True:
9         command = input(">>> ").lower()
10        if command == 'exit':
11            break
12
13        elif command == 'add':
14            class_name = input("Введите класс ")
15            pupils = int(input("Введите кол-во учащихся в данном классе "))
16            school[class_name] = pupils
17
18        elif command == 'list':
19            line = '+-{}-+-{}-+-{}-+'.format(
20                '-' * 4,
21                '-' * 30,
22                '-' * 20,
23            )
24            print(line)
25            print(
26                '| {:^4} | {:^30} | {:^20} |'.format(
27                    "No",
28                    "Класс",
29                    "Количество учеников",
30                )
31            )
32            print(line)
33            index = 0
34            for class_name, pupils in school.items():
35                index += 1
36                print(
37                    '| {:>4} | {:<30} | {:<20} |'.format(
38                        index,
39                        class_name,
40                        pupils,
41                    )
42                )
43            print(line)
44
45 if __name__ == '__main__': > while True > elif command == 'list'

```

Рисунок 5 – Код программы

```

38         index,
39         class_name,
40         pupils,
41     )
42 )
43 print(line)
44
45 elif command == 'edit':
46     class_name = input("Введите класс, в котором нужно внести "
47                         "изменения ")
48     pupils = input("Введите новое количество учащихся в классе "
49                   f"{class_name} ")
50     school[class_name] = pupils
51     print("Количество учащихся было успешно изменено!")
52
53 elif command == 'delete':
54     class_name = input("Введите класс, который нужно расформировать ")
55     del school[class_name]
56     print("Класс был успешно расформирован")
57
58 elif command == 'pupils':
59     sum_of_pupils = 0
60     for class_name in school:
61         sum_of_pupils += int(school[class_name])
62     print(f"Количество учеников в школе: {sum_of_pupils}")
63
64 elif command == 'help':
65     # Вывести справку о работе с программой.
66     print("Список команд:\n")
67     print("add - добавить класс;")
68     print("list - вывести список классов;")
69     print("edit - изменить количество учащихся в заданном классе.")
70     print("delete - расформировать класс.")
71     print("pupils - вывести общее количество учеников во всех "
72           "классах.")
73     print("help - отобразить справку;")
74     print("exit - завершить работу с программой.")
75 else:
76     print(f"Неизвестная команда {command}", file=sys.stderr)
77

```

Рисунок 6 – Код программы, продолжение

```
>>> add
Введите класс 1A
Введите кол-во учащихся в данном классе 23
>>> add
Введите класс 1Б
Введите кол-во учащихся в данном классе 25
>>> add
Введите класс 9Г
Введите кол-во учащихся в данном классе 30
>>> add
Введите класс 9Д
Введите кол-во учащихся в данном классе 31
>>> add
Введите класс 11А
Введите кол-во учащихся в данном классе 26
>>> list
```

No	Класс	Количество учеников
1	1A	23
2	1Б	25
3	9Г	30
4	9Д	31
5	11А	26

Рисунок 7 – Заполнение словаря

```
>>> edit
Введите класс, в котором нужно внести изменения 9Г
Введите новое количество учащихся в классе 9Г 32
Количество учащихся было успешно изменено!
>>> list
```

No	Класс	Количество учеников
1	1A	23
2	1Б	25
3	9Г	32
4	9Д	31
5	11А	26

Рисунок 8 – Изменение количества учащихся в классе

```
>>> add
Введите класс 11Б
Введите кол-во учащихся в данном классе 19
>>> list
```

No	Класс	Количество учеников
1	1А	23
2	1Б	25
3	9Г	32
4	9Д	31
5	11А	26
6	11Б	19

Рисунок 9 – Добавление нового класса

```
>>> delete
Введите класс, который нужно расформировать 9Д
Класс был успешно расформирован
>>> list
```

No	Класс	Количество учеников
1	1А	23
2	1Б	25
3	9Г	32
4	11А	26
5	11Б	19

Рисунок 10 – Расформирование класса

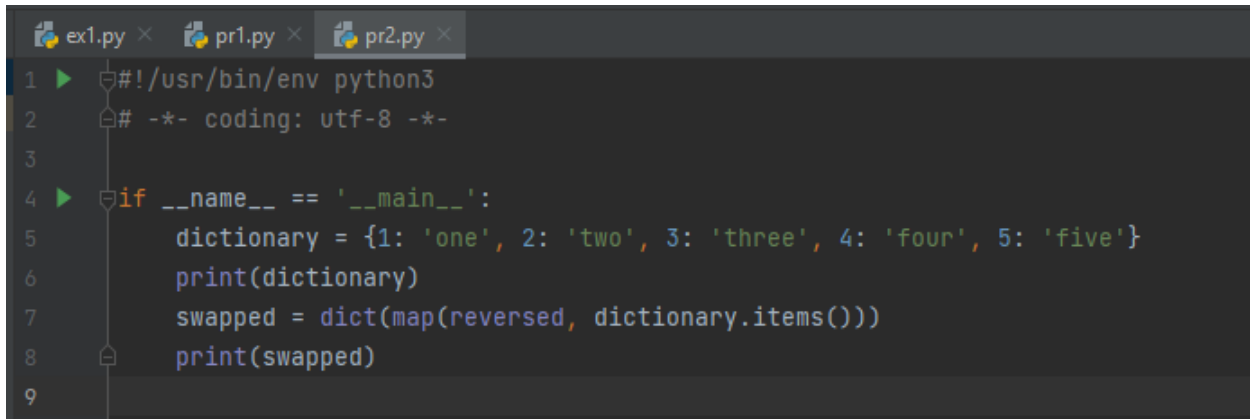
```
>>> pupils
Количество учеников в школе: 125
```

Рисунок 11 – Вывод общего количества учеников в школе

```
>>> iljh
>>> Неизвестная команда iljh
```

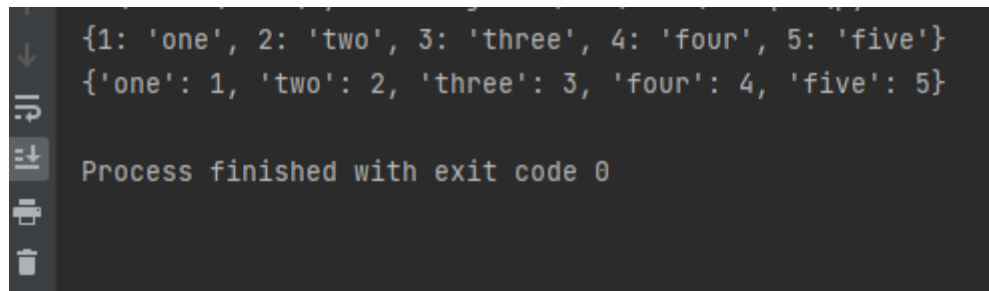
Рисунок 12 – Вывод программы при неправильной команде

1.3 Задача 2 (рис 13, 14).



```
ex1.py x pr1.py x pr2.py x
1  > #!/usr/bin/env python3
2  > # -*- coding: utf-8 -*-
3
4  > if __name__ == '__main__':
5      dictionary = {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
6      print(dictionary)
7      swapped = dict(map(reversed, dictionary.items()))
8      print(swapped)
9
```

Рисунок 13 – Код программы



```
{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 14 – Вывод программы

1.4 Индивидуальное задание №1 (рис. 15, 16, 17, 18).

```

#!/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', ''))

```

Рисунок 15 – Код программы


```

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)

    for idx, flight in enumerate(flights, 1):
        print(
            '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
                idx,
                flight.get('flight_destination', ''),
                flight.get('flight_number', ''),
                flight.get('airplane_type', 0)
            )
        )
    print(line)

```

Рисунок 16 – Код программы, продолжение

```

elif command.startswith('select '):
    parts = command.split(' ', maxsplit=1)
    airplane_type = (parts[1].capitalize())
    print(f"Для типа самолета {airplane_type}:")
    count = 0
    for flight in flights:
        if flight.get('airplane_type') == airplane_type:
            count += 1
            print(
                '{:>4}: Пункт назначения: {}; Номер рейса: {}'.format(
                    count,
                    flight.get('flight_destination',
                                ''),
                    flight.get('flight_number', ''))
            )
    if count == 0:
        print("рейсы не найдены")

elif command == 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить рейс;")
    print("list - вывести список всех рейсов;")
    print("select <тип самолета> - запросить рейсы указанного типа "
          "самолета;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
else:
    print(f"Неизвестная команда {command}", file=sys.stderr)

```

Рисунок 17 – Код программы, продолжение

```

>>> add
Введите название пункта назначения Москва
Введите номер рейса 5
Введите тип самолета FX
>>> list
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолета |
+-----+-----+-----+-----+
| 1 | Москва | 5 | FX |
+-----+-----+-----+-----+
>>> |

```

Рисунок 18 – Результат выполнения

2. 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`:

1. Класс `datetime.date(year, month, day)` - стандартная дата.
Атрибуты: `year`, `month`, `day`. Неизменяемый объект.
2. 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`.
3. Класс `datetime.timedelta` - разница между двумя моментами времени, с точностью до микросекунд.
4. Класс `datetime.tzinfo` - абстрактный базовый класс для информации о временной зоне (например, для учета часового пояса и / или летнего времени).
5. 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)`
- $1 \leq \text{month} \leq 12$
- $1 \leq \text{day} \leq \text{количество дней в данном месяце и году}$

Необязательные:

- $0 \leq \text{minute} < 60$
- $0 \leq \text{second} < 60$
- $0 \leq \text{microsecond} < 1000000$

Методы класса `datetime`:

1. `datetime.today()` - объект `datetime` из текущей даты и времени.
Работает также, как и `datetime.now()` со значением `tz=None`.
2. `datetime.fromtimestamp(timestamp)` - дата из стандартного представления времени.
3. `datetime.fromordinal(ordinal)` - дата из числа, представляющего собой количество дней, прошедших с 01.01.1970.
4. `datetime.now(tz=None)` - объект `datetime` из текущей даты и времени.
5. `datetime.combine(date, time)` - объект `datetime` из комбинации объектов `date` и `time`.
6. `datetime.strptime(date_string, format)` - преобразует строку в `datetime` (так же, как и функция `strptime` из модуля `time`).
7. `datetime.strptime(format)` - см. функцию `strptime` из модуля `time`.
8. `datetime.date()` - объект даты (с отсечением времени).
9. `datetime.time()` - объект времени (с отсечением даты).

10. `datetime.replace([year[, month[, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]]])` - возвращает новый объект `datetime` с изменёнными атрибутами.
11. `datetime.timetuple()` - возвращает `struct_time` из `datetime`.
12. `datetime.toordinal()` - количество дней, прошедших с 01.01.1970.
13. `datetime.timestamp()` - возвращает время в секундах с начала эпохи.
14. `datetime.weekday()` - день недели в виде числа, понедельник - 0, воскресенье - 6.
15. `datetime.isoweekday()` - день недели в виде числа, понедельник - 1, воскресенье - 7.
16. `datetime.isocalendar()` - кортеж (год в формате ISO, ISO номер недели, ISO день недели).
17. `datetime.isoformat(sep='T')` - красивая строка вида "YYYY-MM-DDTHH:MM:SS.mmmmmm" или, если `microsecond == 0`, "YYYY-MM-DDTHH:MM:SS"
18. `datetime.ctime()` - преобразует время, выраженное в секундах с начала эпохи в строку вида "Thu Sep 27 16:42:37 2012".