

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

**Отчёт о лабораторной работе №9 по дисциплине основы программной
инженерии**

Выполнила:

Нестеренко Тамара Антоновна,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ

1. Практическая часть

```
>>> a = {'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь'}
>>> a
{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь'}
```

Рисунок 1 – Пример ввода словаря

```
>>> a['cat']
'кошка'
>>> a['bird']
'птица'
>>>
```

Рисунок 2 – Пример доступа к значениям словаря по ключам

```
>>> a['elephant'] = 'бегемот'
>>> a['table'] = 'стол'
>>> a
{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь', 'elephant': 'бегемот', 'table': 'стол'}
>>> a['elephant'] = 'слон'
>>> del a['table']
>>> a
{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь', 'elephant': 'слон'}
>>>
```

Рисунок 3 – Пример работы со словарём: добавление, изменение, удаление

```
>>> nums = {1: 'one', 2: 'two', 3: 'three'}
>>> person = {'name': 'Tom', 1: [30, 15, 16], 2: 2.34, ('ab', 100): 'no'}
>>>
```

Рисунок 4 – Пример различных структур словаря

```
>>> nums
{1: 'one', 2: 'two', 3: 'three'}
>>> for i in nums:
...     print(i)
...
1
2
3
>>>
```

Рисунок 5 – Пример перебора элементов словаря в цикле for, извлечение ключей

```
>>> for i in nums:
...     print(nums[i])
...
one
two
three
>>>
```

Рисунок 6 – Пример перебора элементов словаря в цикле for, извлечение значений

```
>>> n = nums.items()
>>> n
dict_items([(1, 'one'), (2, 'two'), (3, 'three')])
>>>
```

Рисунок 7 – Пример работы метода items()

```
>>> for key, value in nums.items():
...     print(key, 'is', value)
...
1 is one
2 is two
3 is three
>>>
```

Рисунок 8 – Пример распаковки кортежа

```
>>> v_nums = []
>>> for v in nums.values():
...     v_nums.append(v)
...
>>> v_nums
['one', 'two', 'three']
>>>
```

Рисунок 9 – Пример работы методов key() и value()

```
>>> a
{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь', 'elephant': 'слон'}
>>> a.clear()
>>> a
{}
>>>
```

Рисунок 10 – Пример работы метода clear()

```

>>> nums2 = nums.copy()
>>> nums2[4] = 'four'
>>> nums
{1: 'one', 2: 'two', 3: 'three'}
>>> nums2
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
>>>

```

Рисунок 11 – Пример работы метода copy()

```

>>> a = [1, 2, 3]
>>> c = dict.fromkeys(a)
>>> c
{1: None, 2: None, 3: None}
>>> d = dict.fromkeys(a, 10)
>>> d
{1: 10, 2: 10, 3: 10}
>>> c
{1: None, 2: None, 3: None}
>>>

```

Рисунок 12 – Пример работы метода fromkeys()

```

>>> nums.get(1)
'one'
>>>

```

Рисунок 13 – Пример работы метода get()

```

>>> nums.pop(1)
'one'
>>> nums
{2: 'two', 3: 'three'}
>>> nums.popitem()
(3, 'three')
>>> nums
{2: 'two'}
>>>

```

Рисунок 14 – Пример работы методов pop() и popitem()

```

>>> nums.setdefault(4, 'four')
'four'
>>> nums
{2: 'two', 4: 'four'}
>>>

```

Рисунок 15 – Пример работы метода setdefault()

```
>>> nums.update({6: 'six', 7: 'seven'})
>>> nums
{2: 'two', 4: 'four', 6: 'six', 7: 'seven'}
>>>
```

Рисунок 16 – Пример работы метода update()

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

Рисунок 17 – Пример создание словаря включений

```
>>> {name: len(name) for name in ('Stack', 'Overflow', 'Exchange') if len(name) > 6}
{'Overflow': 8, 'Exchange': 8}
>>> dict((name, len(name)) for name in ('Stack', 'Overflow', 'Exchange') if len(name) > 6)
{'Overflow': 8, 'Exchange': 8}
>>>
```

Рисунок 18 – Пример использования условного оператора внутри словаря включения и использования генерального выражения

```
>>> initial_dict = {'x': 1, 'y': 2}
>>> {key: value for key, value in initial_dict.items() if key == 'x'}
{'x': 1}
```

Рисунок 19 – Пример использования словаря в качестве фильтра пары ключ-значение

```
>>> my_dict = {1: 'a', 2: 'b', 3: 'c'}
>>> swapped = {v: k for k, v in my_dict.items()}
>>> swapped = dict((v, k) for k, v in my_dict.items())
>>> swapped = dict(zip(my_dict.values(), my_dict))
>>> swapped = dict(zip(my_dict.values(), my_dict.keys()))
>>> swapped = dict(map(reversed, my_dict.items()))
>>> print(swapped)
{'a': 1, 'b': 2, 'c': 3}
>>>
```

Рисунок 20 – Пример инвертирования словаря

```
>>> dict1 = {'w': 1, 'x': 1}
>>> dict2 = {'x': 2, 'y': 2, 'z': 2}
>>> {k: v for d in [dict1, dict2] for k, v in d.items()}
{'w': 1, 'x': 2, 'y': 2, 'z': 2}
>>>
```

Рисунок 21 – Пример объединения словаря

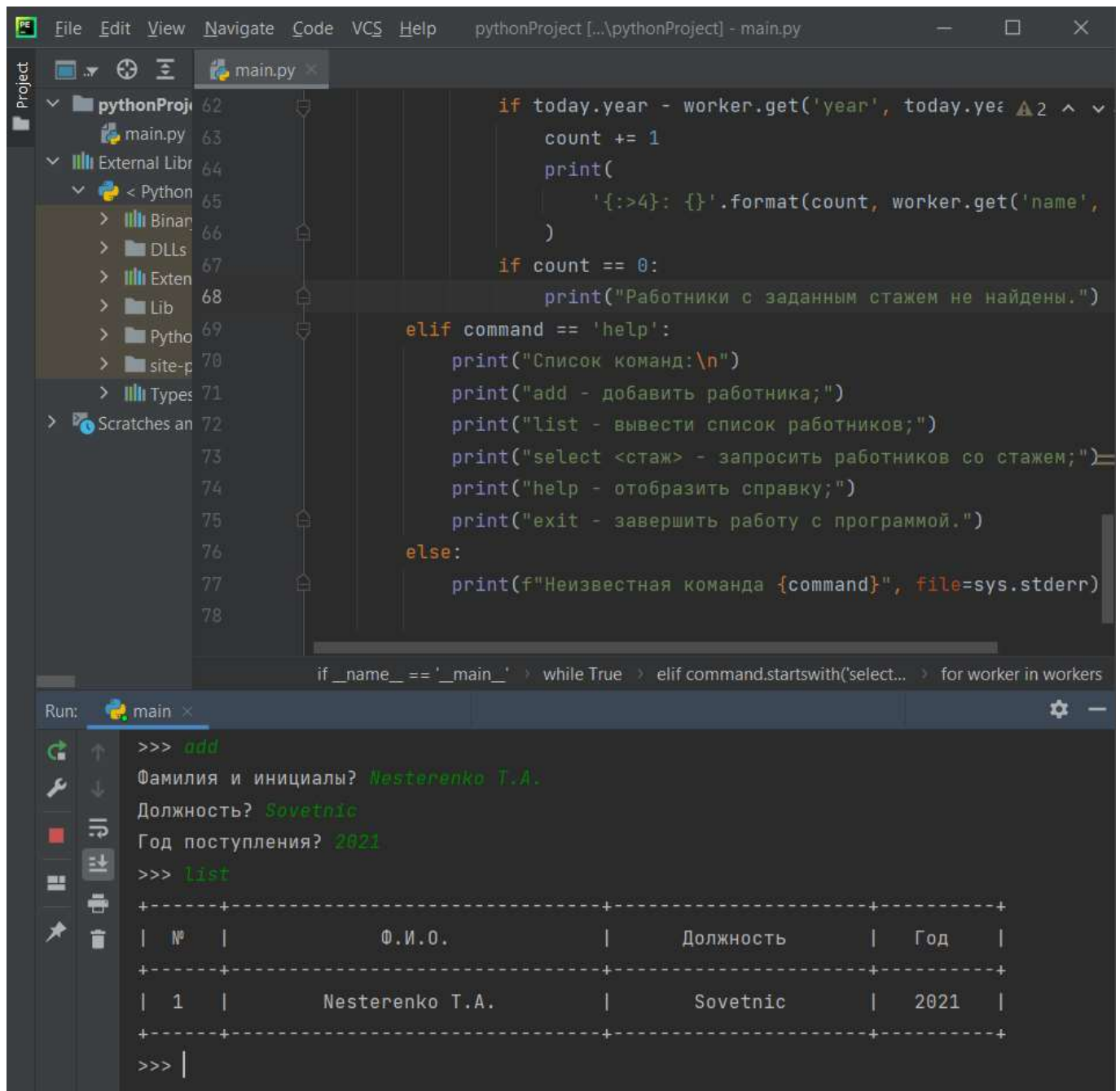


Рисунок 22 – Пример работы примера

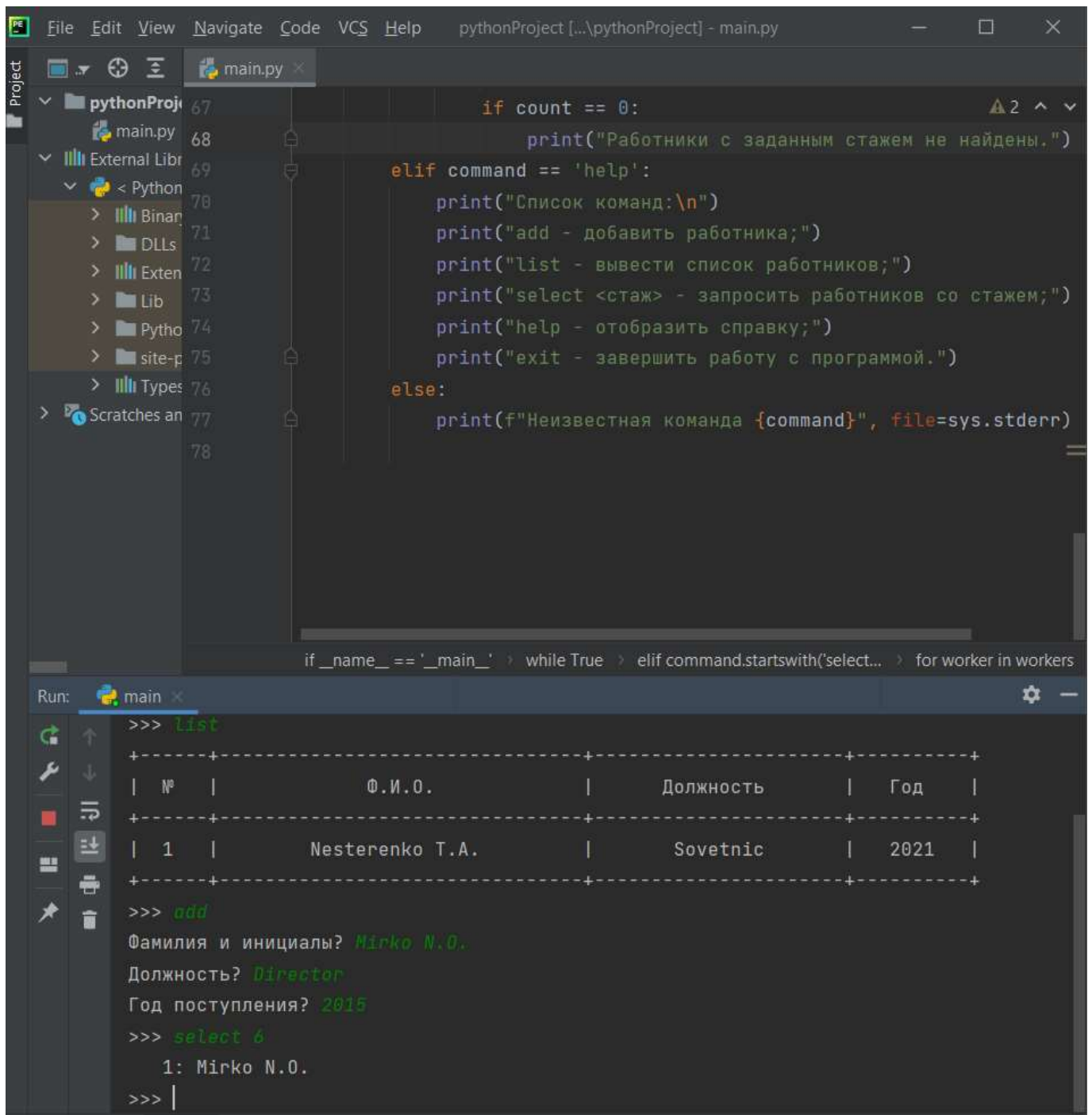


Рисунок 23 – Пример работы примера

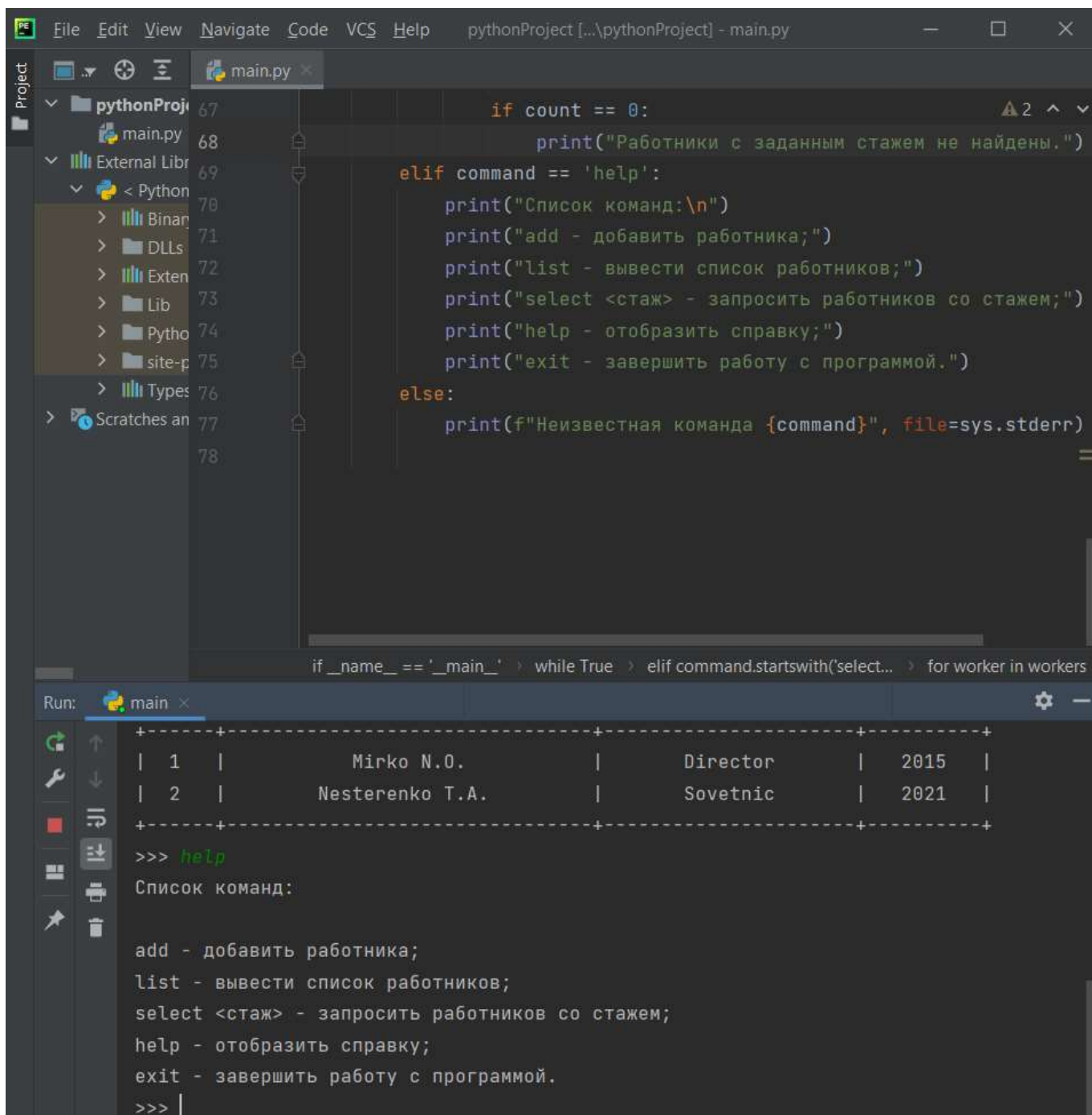
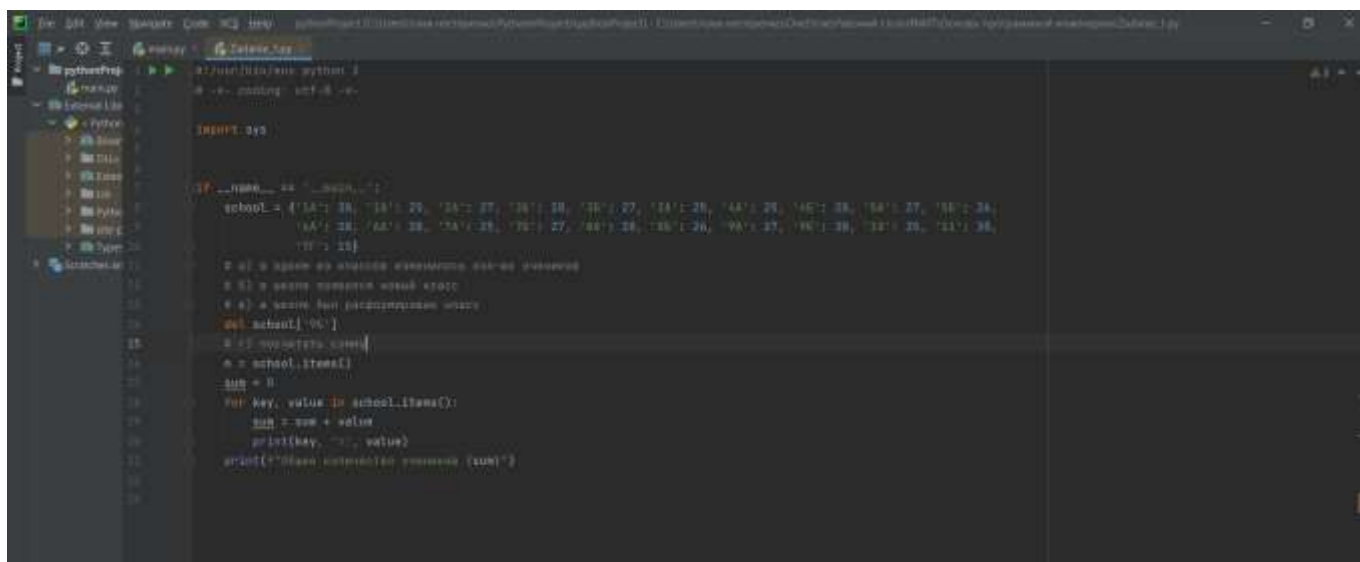


Рисунок 24 – Пример работы примера



The screenshot shows the PyCharm IDE with a project named 'pythonProject'. The file explorer on the left shows a directory structure with 'pythonProject' and 'External Libs'. The main editor displays a file named 'Zadanie_1.py' with the following code:

```
#!/usr/bin/env python 3
# -*- coding: utf-8 -*-

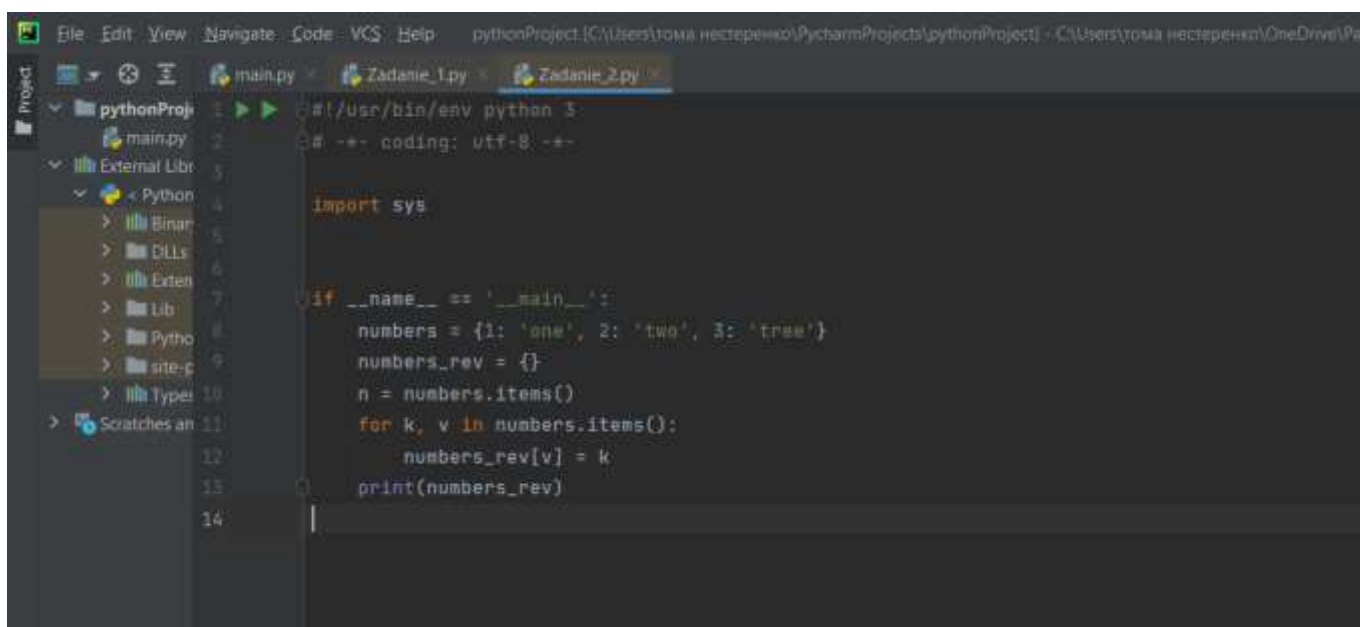
import sys

if __name__ == '__main__':
    school = {'1A': 38, '1B': 29, '1C': 27, '2A': 30, '2B': 27, '3A': 28, '4A': 26, '4B': 38, '5A': 27, '5B': 24,
              '6A': 28, '6B': 38, '7A': 28, '7B': 27, '8A': 26, '8B': 26, '9A': 27, '9B': 38, '10A': 28, '11A': 36,
              '12A': 24}

    # a) в школе по каждому классу есть свой кабинет
    # b) в школе по каждому классу есть свой этаж
    # c) в школе для каждого класса есть свой кабинет и этаж
    del school['9A']

    # d) пометить класс
    n = school.items()
    sum = 0
    for key, value in school.items():
        sum = sum + value
    print(key, ":", value)
    print("Общая стоимость помещений (sum)")
```

Рисунок 25 – Пример решения задания



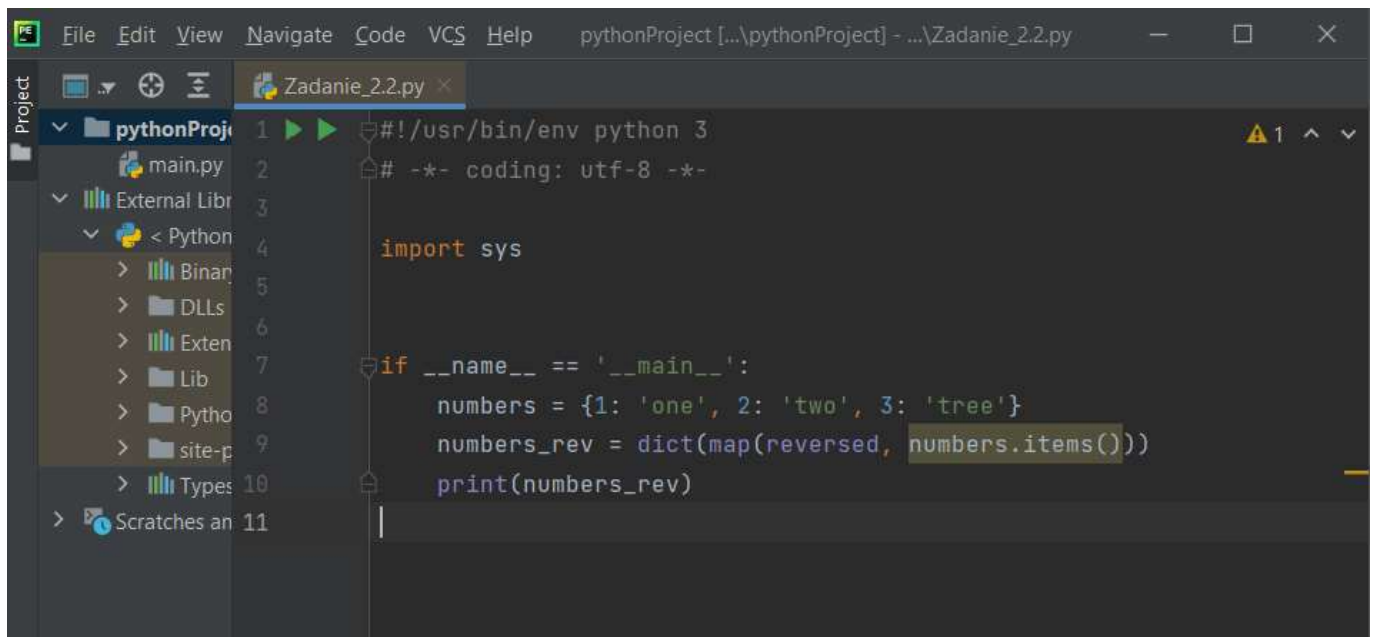
The screenshot shows the PyCharm IDE with a project named 'pythonProject'. The file explorer on the left shows a directory structure with 'pythonProject' and 'External Libs'. The main editor displays a file named 'Zadanie_2.py' with the following code:

```
#!/usr/bin/env python 3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    numbers = {1: 'one', 2: 'two', 3: 'tree'}
    numbers_rev = {}
    n = numbers.items()
    for k, v in numbers.items():
        numbers_rev[v] = k
    print(numbers_rev)
```

Рисунок 26 – Пример решения задания



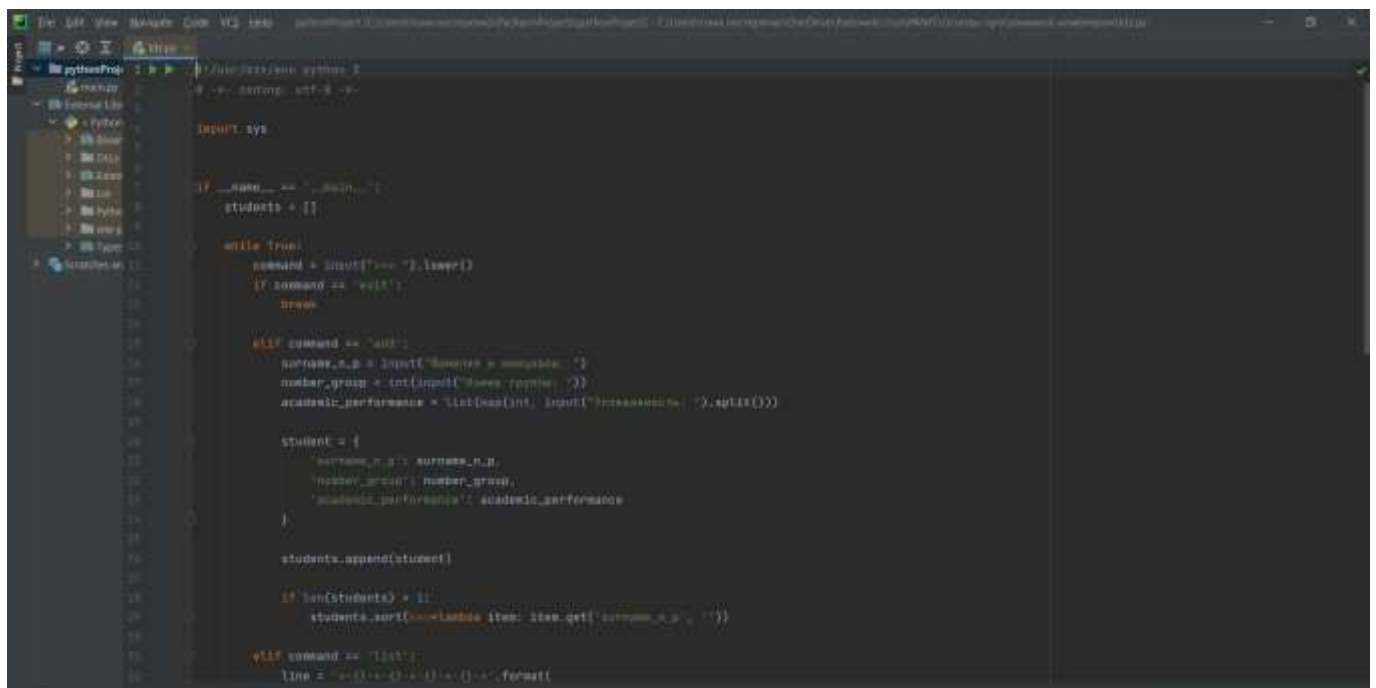
The screenshot shows an IDE window titled 'pythonProject [...\pythonProject] - ...\Zadanie_2.2.py'. The left sidebar displays a project tree with folders like 'pythonProject', 'main.py', 'External Libraries', and 'Python'. The main editor area shows the following Python code:

```
#!/usr/bin/env python 3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    numbers = {1: 'one', 2: 'two', 3: 'tree'}
    numbers_rev = dict(map(reversed, numbers.items()))
    print(numbers_rev)
```

Рисунок 27 – Пример решения задания



The screenshot shows an IDE window titled 'pythonProject [...\pythonProject] - ...\Zadanie_2.2.py'. The left sidebar displays a project tree with folders like 'pythonProject', 'main.py', 'External Libraries', and 'Python'. The main editor area shows the following Python code:

```
#!/usr/bin/env python 3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    students = []

    while True:
        command = input("Enter command: ")
        if command == "exit":
            break

        elif command == "add":
            surname_n_p = input("Surname, name, patronymic: ")
            number_group = int(input("Group number: "))
            academic_performance = list(map(int, input("Academic performance: ").split()))

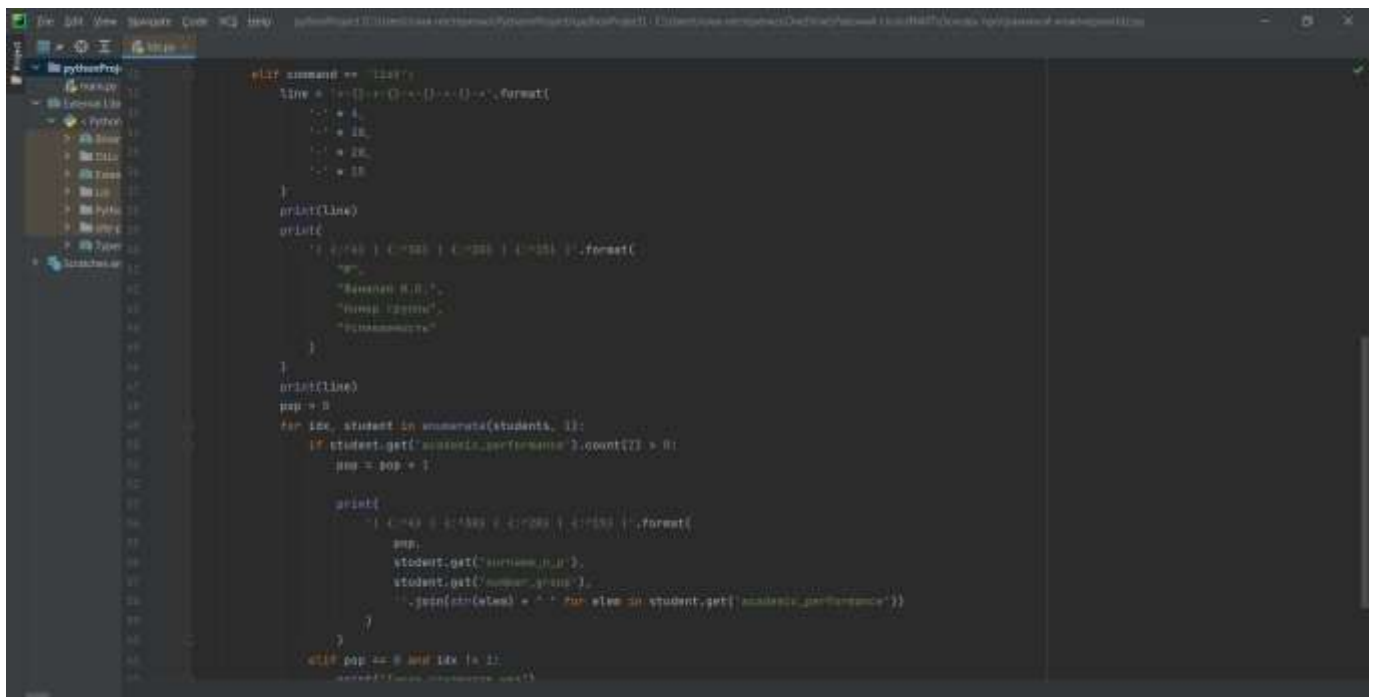
            student = {
                'surname_n_p': surname_n_p,
                'number_group': number_group,
                'academic_performance': academic_performance
            }

            students.append(student)

        if len(students) > 1:
            students.sort(key=lambda item: item.get('surname_n_p', ''))

        elif command == "list":
            line = "%-15s %-10s %-10s" % ("", "Number", "Performance")
            for student in students:
```

Рисунок 28 – Пример решения индивидуального задания



```
elif command == "list":
    line = "{:4s} {:10s} {:10s} {:10s} {}".format(
        "ID", "Name", "Group", "Performance", ""
    )
    print(line)
    print()
    for idx, student in enumerate(students, 1):
        if student.get("academic_performance")["point"] > 0:
            pop = pop + 1

            print(
                "{:4s} {:10s} {:10s} {:10s} {}".format(
                    pop,
                    student.get("surname_n_p"),
                    student.get("number_group"),
                    student.get("academic_performance")["point"],
                    ""
                )
            )
    elif pop == 0 and idx != 1:
        print("There are no students")
    print(line)
```

Рисунок 29 – Пример решения индивидуального задания



```
print(
    "{:4s} {:10s} {:10s} {:10s} {}".format(
        pop,
        student.get("surname_n_p"),
        student.get("number_group"),
        student.get("academic_performance")["point"],
        ""
    )
)
elif pop == 0 and idx != 1:
    print("There are no students")
print(line)
```

Рисунок 30 – Пример решения индивидуального задания

2. Вопросы для защиты работы

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.

Словарь - это изменяемый (как список) неупорядоченный (в отличие от строк, списков и кортежей) набор элементов "ключ: значение".

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

Да, len() возвращает целое число, представляющее количество пар key:value в словаре.

3. Какие методы обхода словарей Вам известны?

```
>>> nums
{1: 'one', 2: 'two', 3: 'three'}
>>> for i in nums:
...     print(i)
...
1
2
3
```

1.

```
for value in currencies.values():
    print(value)
# 1
# 69.78
# 78.28
# 0.65
```

2.

3. Какими способами можно получить значения из словаря по ключу?

```
>>> for i in nums:
...     print(nums[i])
...
one
two
three
```

1.

```
>>> for key, value in nums.items():
...     print(key, 'is', value)
...
1 is one
2 is two
3 is three
```

2.

4. Какими способами можно установить значение в словаре по ключу?

```
>>> v_nums = []
>>> for v in nums.values():
...     v_nums.append(v)
...
>>> v_nums
['one', 'two', 'three']
```

1.

```
>>> for key, value in nums.items():
...     print(key, 'is', value)
...
1 is one
2 is two
3 is three
```

2.

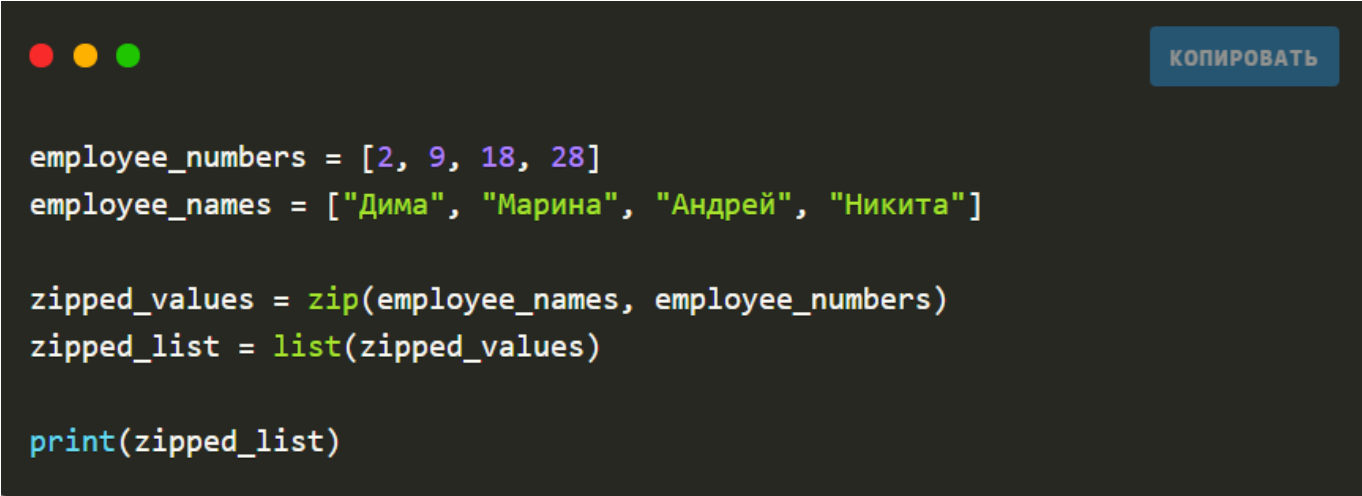
5. Что такое словарь включений?

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}
```

6. Самостоятельно изучите возможности функции zip() приведите примеры её использования.

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.



```
employee_numbers = [2, 9, 18, 28]
employee_names = ["Дима", "Марина", "Андрей", "Никита"]

zipped_values = zip(employee_names, employee_numbers)
zipped_list = list(zipped_values)

print(zipped_list)
```

Функция zip возвращает следующее:

```
[('Дима', 2), ('Марина', 9), ('Андрей', 18), ('Никита', 28)]
```

[КОПИРОВАТЬ](#)

```
employee_numbers = [2, 9, 18, 28]
employee_names = ["Дима", "Марина", "Андрей", "Никита"]

for name, number in zip(employee_names, employee_numbers):
    print(name, number)
```

Этот код вернет следующее:

```
[('Дима', 2), ('Марина', 9), ('Андрей', 18), ('Никита', 28)]
```

[КОПИРОВАТЬ](#)

```
employees_zipped = [('Дима', 2), ('Марина', 9), ('Андрей', 18), ('Никита', 28)]
employee_names, employee_numbers = zip(*employees_zipped)

print(employee_names)
print(employee_numbers)
```

Этот код вернет такой результат:

```
("Дима", "Марина", "Андрей", "Никита")
(2, 9, 18, 28)
```

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



Datetime — важный элемент любой программы, написанной на Python. Этот модуль позволяет управлять датами и временем, представляя их в таком виде, в котором пользователи смогут их понимать.

datetime включает различные компоненты. Так, он состоит из объектов следующих типов:

date — хранит дату

time — хранит время

datetime — хранит дату и время





```
import datetime

dt_now = datetime.datetime.now()
print(dt_now)
```

А вот результат:

```
2020-11-14 15:43:32.249588
```



```
from datetime import date

current_date = date.today()
print(current_date)
```

Результат:

```
2020-11-14
```

[КОПИРОВАТЬ](#)

```
import datetime

current_date_time = datetime.datetime.now()
current_time = current_date_time.time()
print(current_time)
```

Результат:

15:51:05.627643

[КОПИРОВАТЬ](#)

```
import datetime

timeobj= datetime.time(8,48,45)
print(timeobj)
```

Результат такой:

08:48:45


```
datetime.datetime(year,month,day))
```

Такой пример:

```
import datetime

date_obj = datetime.datetime(2020,10,17)
print(date_obj)
```

КОПИРОВАТЬ

Вернет вот такой результат:

```
2020-10-17 00:00:00
```

```
from datetime import datetime

datetime_string = "11/17/20 15:02:34"
datetime_obj = datetime.strptime(datetime_string, '%m/%d/%y %H:%M:%S')
print(datetime_obj)
```

КОПИРОВАТЬ

Результат:

```
2020-11-17 15:02:34
```

Ссылка на репозиторий: https://github.com/tamaranesterenko/Python.LR_9