Федеральное государственное автономное образовательное учреждение
высшего образования «Северо-Кавказский федеральный университет»
Институт цифрового развития

ОТЧЕТ ПО ДИСЦИПЛИНЕ
ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

Лабораторная работа №9

Работа со словарями в языке Python

Выполнила:
Мирзаева Камилла Мирзаевна
2 курс, группа ПИЖ-б-о-20-1

Принял:
Воронкин Роман Александрович

Ставрополь, 2021 г.

Ход работы:

```
>>> 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', 't
able': '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 методов key() и value()

```
>>> 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 setdafault()

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

Рисунок 16 – Пример работы метода setdafault()

```
>>> {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

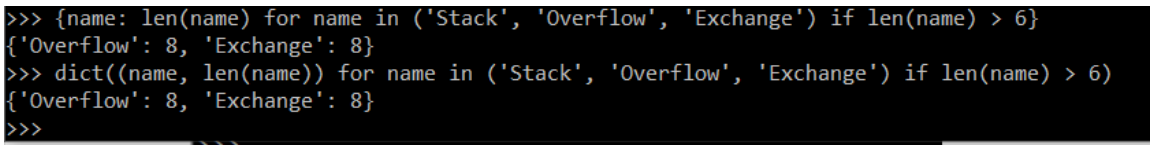


Рисунок 18 – Пример использования условного оператора внутри словаря включения и использования генерального выражения

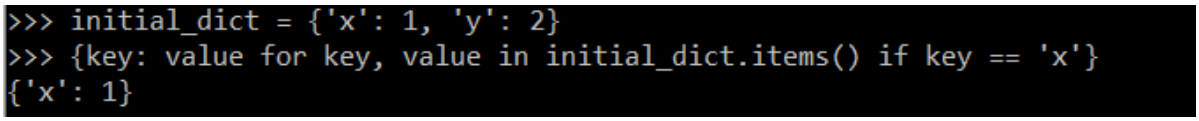


Рисунок 19 – Пример использования словаря в качестве фильтра пары ключ-значение

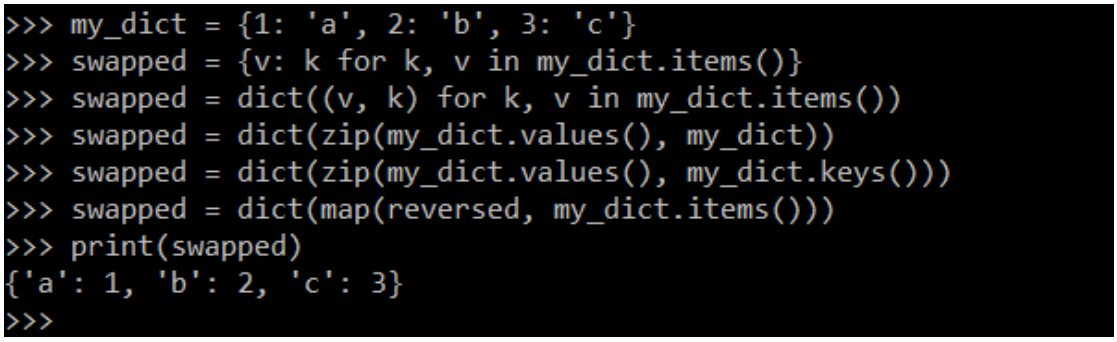


Рисунок 20 – Пример инвертирования словаря

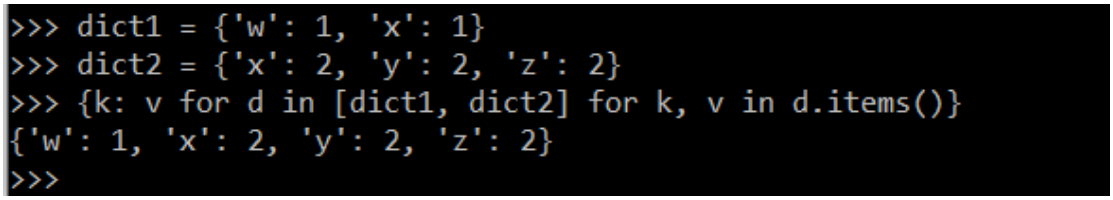
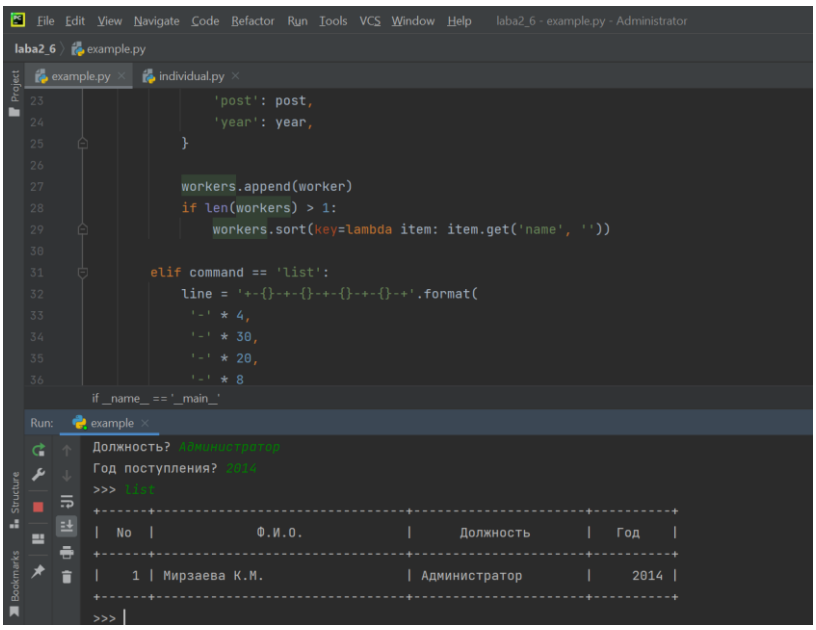


Рисунок 21 – Пример объединения словаря

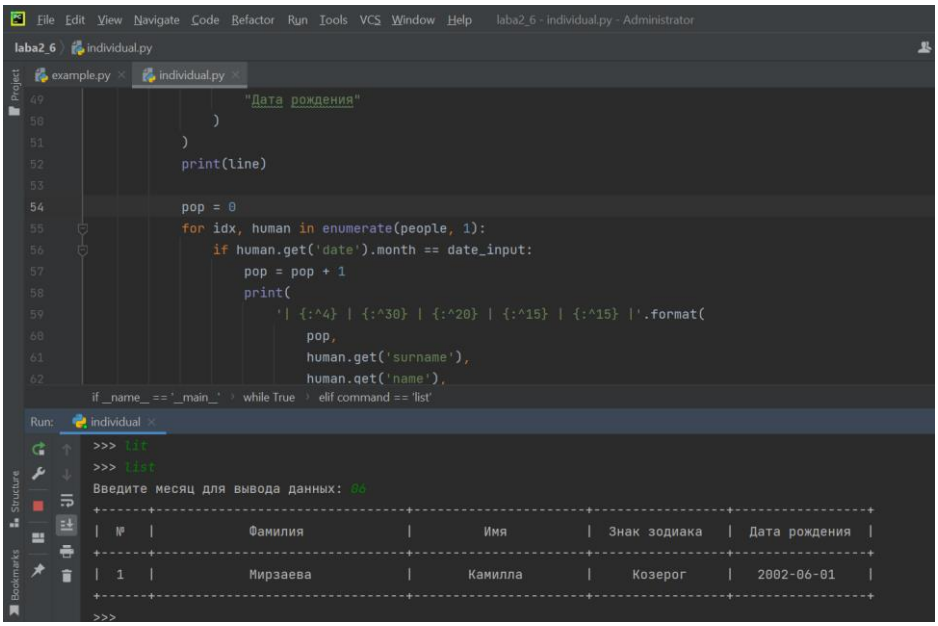
Пример №1



Индивидуальное задание.

Вариант 15.

Использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата рождения (список из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по датам рождения; вывод на экран информации о людях, родившихся под знаком, название которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение.



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.

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?**

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

2.
```
for value in currencies.values():
    print(value)
# 1
# 69.78
# 78.28
# 0.65
```

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

1.
```
>>> for i in nums:
...     print(nums[i])
...
one
two
three
```

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

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

1.
```
>>> v_nums = []
>>> for v in nums.values():
...     v_nums.append(v)
...
>>> v_nums
['one', 'two', 'three']
```
2.
```
>>> for key, value in nums.items():
...     print(key, 'is', value)
...
1 is one
2 is two
3 is three
```

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)]
```

```
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)]
```

```
employees_zipped = [('Дима', 2), ('Марина', 9), ('Андрей', 18), ('Никита', 2
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
```

```
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:

КОПИРОВАТЬ

```
import datetime

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

Вернет вот такой результат:

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

КОПИРОВАТЬ

```
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)
```

Результат:

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