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

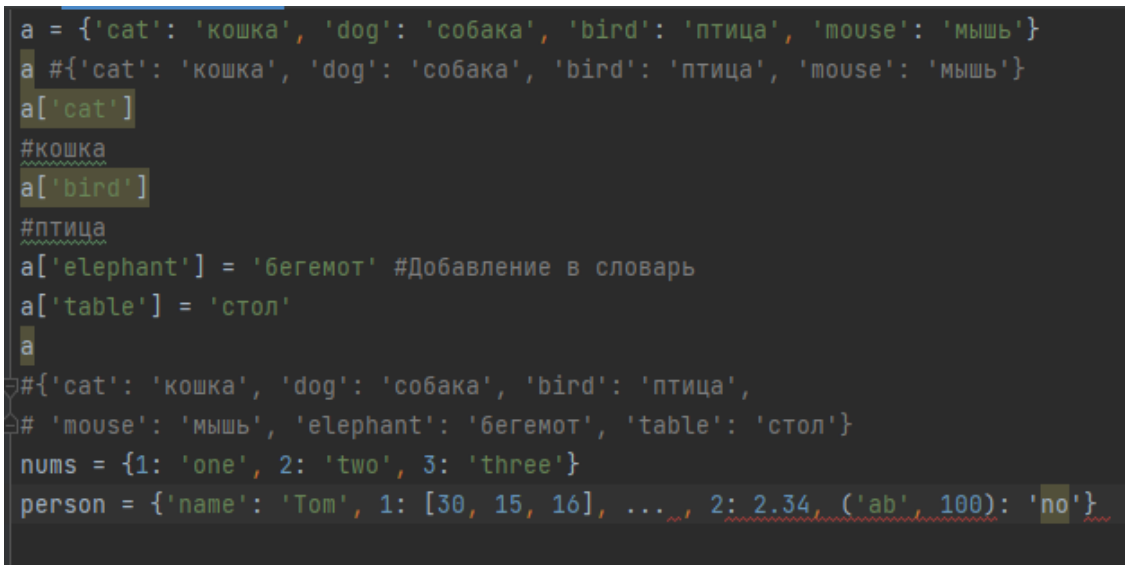
по дисциплине основы программной инженерии

Выполнил: Духно Михаил Александрович,

2 курс, группа ПИЖ-б-о-20-1,

Проверил: Доцент кафедры инфокоммуникаций, Воронкин Р.А.

Ставрополь, 2021 г



```
a = {'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь'}
a #{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь'}
a['cat']
#кошка
a['bird']
#птица
a['elephant'] = 'бегемот' #Добавление в словарь
a['table'] = 'стол'
a
#{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица',
# 'mouse': 'мышь', 'elephant': 'бегемот', 'table': 'стол'}
nums = {1: 'one', 2: 'two', 3: 'three'}
person = {'name': 'Tom', 1: [30, 15, 16], ..., 2: 2.34, ('ab', 100): 'no'}
```

Рисунок 9.1 – Введение в словари

```
nums
#{1: 'one', 2: 'two', 3: 'three'}
for i in nums:
    print(i)
for i in nums:
    print(nums[i])
'''
one
two
three
'''


n = nums.items()
n
#dict_items([(1, 'one'), (2, 'two'), (3, 'three')])
for key, value in nums.items():
    print(key, 'is', value)
'''
is one
is two
is three
'''


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

Рисунок 9.2 – Перебор элементов в цикле

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

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



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}



nums.get(1)
#'one'



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



nums.setdefault(4, 'four')
#'four'
nums
#{3: 'three', 4: 'four'}



nums.update({6: 'six', 7: 'seven'})
#{3: 'three', 4: 'four', 6: 'six', 7: 'seven'}
```

Рисунок 9.3 – Методы словарей

```
{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}



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

```
#{'exchange': 8, 'overflow': 8}



dict((name, len(name)) for name in ('stack', 'overflow', 'exchange') if
len(name) > 6)
#{'exchange': 8, 'overflow': 8}



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



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 9.4 – Словарь включений

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

def add_element():
    name = input('Конечный пункт: ')
    num = input('Номер поезда: ')
    tm = input('Время отправления: ')
    trains = {}
    trains['name'] = name
    trains['num'] = int(num)
    trains['tm'] = tm
    return trains


def find_train(trains):
    num = int(input('Введите номер искомого поезда: '))
    for dcts in trains:
        if dcts['num'] == num:
            print(f'Конечный пункт: {dcts["name"]} \n'
                  f'Номер поезда: {dcts["num"]} \n'
                  f'Время отправления: {dcts["tm"]}')
            return
    print('Поезда с таким номером нет')


if __name__ == '__main__':
    flag = True
    trains = []
    while flag:
        print('1. Добавить новый поезд')
        print('2. Вывести информацию о поезде')
        print('3.Выход из программы')
        com = int(input('введите номер команды: '))
        if com == 1:
```

```
            trains.append(add_element())
        elif com == 2:
            find_train(trains)
        elif com == 3:
            flag = False
```

Рисунок 9.5 – Индивидуальное задание

```
Введите номер искомого поезда: 54
Конечный пункт: Сочи
Номер поезда: 54
Время отправления: 15:00
```

Рисунок 9.5 – Результат работы программы

Контрольные вопросы:

1. Что такое словари в языке Python?

Словари в **Python** - неупорядоченные коллекции произвольных объектов с доступом по ключу. Их иногда ещё называют ассоциативными массивами или хеш-таблицами.

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

Да может, она возвращает количество пар ключ значение

3. Какие методы обхода словарей Вам известны?

При помощи цикла фор

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

dct[<key>]

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

dtc[<key>] = <meaning>

{x: x * x for x in (1, 2, 3, 4)} – Словарь включения

6. Что такое словарь включений?

Словарь включений аналогичен списковым включениям, за исключением того, что он создаёт

объект словаря вместо списка.

7. Самостоятельно изучите возможности функции zip() приведите примеры ее 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)
```
```
[('Дима', 2), ('Марина', 9), ('Андрей', 18), ('Никита', 28)]
```

8. Самостоятельно изучите возможности модуля datetime. Каким функционалом по работе с

датой и временем обладает этот модуль?

Модуль datetime предоставляет классы для обработки времени и даты разными способами.

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-MM-DDTHH:MM:SS.mmmmmm" или, если microsecond == 0, "YYYY-MM-DDTHH:MM:SS"

datetime.ctime() - см. ctime() из модуля time.