Тема 3.
Структуры данных в Python

(продолжение)

Кортежи

Кортеж (tuple) – это упорядоченная неизменяемая последовательность элементов.

Кортеж

- **Кортеж – это список, элементы которого нельзя менять.**
- **Обычно используется в случаях, когда данные постоянны на всем протяжении выполнения программы.**
- **Кортеж заключается в круглые скобки (…)**
- **Кортеж может быть многомерным**

Кортежи

- **Способы создания кортежа:**

1. **с помощью функции**
   `tuple([Последовательность])`

Примеры:

```
b = tuple()                  # b = ()
s = tuple("String")          # s = ('S', 't', 'r', 'i', 'n', 'g')
t = tuple( [1,2,3,4,5] )     # t = (1, 2, 3, 4, 5)
```

Кортежи

- **Способы создания кортежа:**
  2. **Перечислением элементов в круглых скобках**

Примеры:

```
a = ()                  # a = ()
b = (4)                 # b = 4 – Число, не кортеж!!!
s = (5, )               # s = (5, ) – кортеж
t = 12, "23", 4.5       # t = (12, '23', 4.5) – кортеж
v = 54,                 # v = (54, ) – кортеж
```

Кортеж

- Операции над кортежами и методы кортежей аналогичны средствам работы со строками при условии, что они не изменяют содержимое.
- К ним можно отнести:
  - Объединение кортежей
  - Повторение кортежей
  - Извлечение среза
  - Доступ по индексу
  - …

Кортежи

- **Упаковка и распаковка кортежей**
  - **упаковка:**
    - **набор элементов, перечисленных через запятую, упаковываются в один кортеж**

```
t = 12, "23", 4.5      #где t = (12, "23", 4.5)
```

Кортежи

- **Упаковка и распаковка кортежей**
  - **распаковка:**
    - **нескольким переменным присваиваются элементы кортежа (число переменных должно совпадать с количеством значений в кортеже)**

```
t = 12, "23", 4.5
x, y, z = t    #где t - кортеж
a = t          # здесь ошибка,
a, *b = t      # a=12, b = ["23", 4.5]
```

Кортежи

- **Пример.**
- **Дан список случайных целых чисел. Требуется вычислить сумму элементов списка, с учетом увеличения всех элементов кратных 4 в 10 раз.**

Кортежи

- **Пример (программа)**

```
def func(lst_, k):
    for i in range(len(lst_)):
        if lst_[i] % k == 0:
            lst_[i]*=10 # увеличиваем элемент в 10 раз
    return sum(lst_) # возвращаем сумму

lst = [randint(20, 79) for i in range(10)]
print(lst)          # [62, 44, 66, 23, 37, 71, 72, 64, 64, 68]
print(func(lst,4))
print(lst)   # [62, 440, 66, 23, 37, 71, 720, 640, 640, 680]
```

Кортежи

- **Изменяемые объекты передаются в функцию по ссылке (т.е. параметру-переменной присваивается ссылка на уже существующий объект)!**
- **Если важно защитить данные от изменения, то следует в функции передавать неизменяемые объекты, например, вместо списка использовать кортеж.**

Кортежи

- **Пример**

```
def func(lst_, k):
    lst_ = list(lst_) # преобразуем кортеж к списку
    for i in range(len(lst_)):
        if lst_[i] % k == 0:
            lst_[i] *= 10 # редактируем элементы
    return sum(lst_)

lst = tuple([randint(20, 79) for i in range(10)])
print(lst)              # [62, 44, 66, 23, 37, 71, 72, 64, 64, 68]
print(func(lst, 4))
print(lst)              # [62, 44, 66, 23, 37, 71, 72, 64, 64, 68]
```

Кортежи

- **Т.к. кортежи могут содержать списки, также как списки быть вложенными в другие списки.**
- **Списки внутри кортежей можно изменять!**

```
tpl = (1,  10, [45, 64,89])
print(tpl)
tpl[2][1] = "fff"
print(tpl)
```

***Результат*:**

```
(1, 10, [45, 64, 89])

(1, 10, [45, 'fff', 89])
```

Словари

Словарь (dict) – это неупорядоченный изменяемый набор элементов “ключ:значение”.

Словари

- **Словарь (Dictionary) – это отображение между ключами (keys) и значениями (values), при котором ключу однозначно соответствует значение.**
- **В словаре не может быть двух одинаковых ключей.**
- **Синтаксис:**

```
{ключ1: значение1, ключ2: значение2,…}
```

Словари

- **Ключом может быть любой неизменяемый тип данных.**
- **Значением – любой тип данных.**
- **Значения словарей вполне могут быть структурами, например, другими словарями или списками.**
- **Доступ к значениям осуществляется по ключам, которые заключаются в квадратные скобки.**

Словари

- **Способы создания словаря:**

1. **Перечисление элементов словаря**

Примеры:

```
x = {} # пустой словарь
d = {1:'Google', 2:'Mail',3:'Yandex', 4:'Yahoo'}

print(d[3])      # ‘Yandex'
```

Словари

- **Способы создания словаря:**
  2. **С помощью функции dict()**

  Примеры:

```
d1 = dict(g='Google',m='Mail', y='Yandex', yl='Yahoo')
print(d1['m'])                    # 'Mail'

d2 = dict ( [ (1, 'Google'), (2, 'Mail'), (3, 'Yandex' ) , (4,'Yahoo')] )
```

Словари

- **Способы создания словаря:**

  3. **С помощью метода fromkeys()**

Примеры:

```
d = dict.fromkeys(['a','b'])
print(d)                        # {'a': None, 'b': None}


d = dict.fromkeys(['a','b'], 100)
print(d)                        # {'a': 100, 'b': 100}
```

Словари

- **Способы создания словаря:**

4. **С помощью генератора словарей**

Синтаксис:

{ ключ: выражение for переменная in набор }

Примеры:

```
d = { a: a ** 2 for a in range(7)}
print(d)                    # {1: 1, 2: 4, 3: 9, 4: 16}
```

Словари

- **Операции над словарями**
  - ***Добавление элементов*** **:**

    для добавления элемента в словарь достаточно присвоить значение по новому ключу.

```
d = { a: a ** 2 for a in range(1,5)}
print(d)          # {1: 1, 2: 4, 3: 9, 4: 16}
d[6]=678
print(d)          # {1: 1, 2: 4, 3: 9, 4: 16, 6: 678}
```

Словари

- Методы словарей
  - dict.clear() – очищает список
  - dict.copy() – возвращает копию словаря
  - dict.get(key[, default]) – возвращает значение ключа (если ключа нет, то возвращает default)
  - dict.items() – возвращает пары (ключ, значение)
  - dict.keys() – возвращает ключи в словаре
  - dict.values() – возвращает значения в словаре

Словари

- **Методы словарей**
  - **dict.pop(key[, default]) – удаляет ключ и возвращает значение (default – по умолчанию)**
  - **dict.popitem() – удаляет и возвращает пару (ключ, значение). Если словарь пуст, исключение KeyError**

Словари

- **Методы словарей**
  - **dict.setdefault(key[, default]) – возвращает значение по ключу (если ключа нет, то создается новая пара key: default)**
  - **dict.update([other]) – обновляет словарь, добавляя пары (ключ, значение) из другого словаря other. Существующие значения перезаписываются. Результат None.**

Словари

- **Перебор элементов словаря в цикле for**
  - **Перебор ключей словаря циклом for :**

```
d = { a: a ** 2 for a in range(1,5) }

for x in d: # перебор ключей словаря d
    print(x, d[x])
# здесь x – ключ, d[x] – значение по ключу
```

Словари

- **Перебор элементов словаря в цикле for**
  - **Перебор пар циклом for по набору items() :**

```
d = { a: a ** 2 for a in range(1,5) }

for key, value in d.items(): # перебор пар
    print(key, value)
```

Словари

- **Пример**
  - **В файле записана информация о результатах олимпиады школьников в виде:**

    фио школа место

  - **Определить по каждой школе общее число участников**

файл

olimp_results

```
Иванов школа_30 1
Петров школа_25 2
Сидоров школа_34 3
Краснов школа_30 3
Кринов школа_32 1
Варов школа_30 2
Жданов школа_33 5
Силуанов школа_33 4
Авернов школа_25 6
```

Словари

- **Пример**

```
f = open("files/olimp_results", encoding="UTF-8")
d = {} # создаем словарь для хранения школ (ключи) и
       # количества участников (значения)
for x in f.readlines():
    info = x.split()                          # считываем значение по школе или
    v = d.setdefault(info[1], 0)              # создаем новую пару
    d[info[1]] = v + 1    # увеличиваем счетчик по школе

for k,v in d.items():
    print(k,"\t",v)
```

Множества

Множество (set) – это неупорядоченная изменяемая последовательность уникальных элементов.

Множество

- ***Множество*** **– это неупорядоченная последовательность уникальных элементов.**
- **Множество может быть как изменяемым (set()), так и неизменяемым (frozenset())**
- **Элементы множества должны быть хешируемыми.**

- **Синтаксис:**

{ элемент1, элемент2, …, элементN }

Множество

- **Способы создания множества:**

1. **Перечисление элементов множества**

Примеры:

d = {1, 2, 3, 4}

***Важно*: пустое множество нельзя создать посредством { }, т.к. в этом случае это будет не множество, а словарь!**

Множество

- **Способы создания множества:**
  2. **С помощью функции set(набор)**

  Примеры:

```
d1 = set()  # пустое множество
d2 = set ( ‘abracadabra')      # {'b', 'd', 'c', 'r', 'a'}
```

Множество

- **Способы создания множества:**

  3. **С помощью генератора множеств**

Синтаксис:

{ выражение for переменная in набор }

Примеры:

```
s = { i**2 for i in range(10) }
print(s)    # {0, 1, 64, 4, 36, 9, 16, 49, 81, 25}
```

Множество

- **Пример.**
- **Определить, сколько различных символов встречается в заданной строке**

```
st=input("Input string: ")
dif_chars=set(list(st))

print(f"set = {dif_chars}\n len={len(dif_chars)}\n")
```

Множество

- **Операции над множествами:**
  - **Объединение множеств: |**
  - **Пересечение множеств: &**
  - **Разность множеств: –**
  - **Симметричная разность: ^**
  - **Сравнение множеств:**
    - **> (включение первого во второе), < (наоборот)**
    - **== (эквивалентность), != (неэквивалентность)**

Множество

- **Операции над множествами:**

```
s1 = set('12345')       # {'3', '2', '4', '1', '5'}
s2 = set('6789345')     # {'3', '4', '8', '5', '6', '7', '9'}
print(s1, s2)
```

```
s1 |  s2 = {'3', '2', '4', '1', '8', '5', '6', '7', '9'}
s1 &  s2 = {'3', '5', '4'}
s1 –  s2 = {'2', '1'}
s1 ^  s2 = {'2', '1', '8', '6', '7', '9'}
```

Множество

- **Пример.**
- **Даны два списка чисел. Выведите все общие элементы списков по одному разу в порядке убывания**

```
l1 = [randint(10, 50) for n in range(100)]
l2 = [randint(10, 100) for n in range(50)]

print(f"l1={l1}\nl2={l2}\n")
print(f"l1&l2 ->  {sorted(list(set(l1)&set(l2)), reverse=True)}")
```

Множество

- **Методы множеств:**
  - **set.union(множества) – объединение**
  - **set.intersection() - пересечение**
  - **set.difference() - разность**
  - **set.symmetric_difference() – симметричная разность**

***Отличия*: аргументы необязательно множества!**

Множество

- **Пример.**

```
st_1 = {3, 7, 8, 89}
st_2 = {4, 100, 8}
words = 'string spring'
lst = [4, 4, 3, 55, 2]
```

```
# Метод union()
print(st_1.union(st_2)) # {3, 100, 4, 7, 8, 89}
print(st_1.union(words)) # {'g', 3, 'n', 7, 8, 't', ' ', 'r', 'i', 'p', 's', 89}
print(st_1.union(lst)) # {2, 3, 4, 7, 8, 55, 89}
# Через оператор |
print(st_1 | st_2) # {3, 100, 4, 7, 8, 89}
print(st_1 | words) # Ошибка TypeError
```

Множество

- **Методы множеств:**
  - **set.isdisjoin(other) – истина, если set и other не имеют общих элементов**
  - **set.issubset(other) – истина, если все элементы set содержаться в other (аналогично, set <= other)**
  - **set.issuperset(other) – истина, если все элементы other содержаться в set (аналогично, set >= other)**

Множество

- Методы множеств (меняют исходное множество):
  - set.add(элемент) – добавляет элемент в множество
  - set.remove(элемент) – удаляет элемент из множества. Если элемента нет, ошибка!
  - set.discard(элемент) – удаляет элемент из множества, если он там есть (без ошибки).

Множество

- Методы множеств (меняют исходное множество) :
  - set.clear() – очищает множество
  - set.pop() – удаляет произвольный элемент из множества и возвращает его в качестве результата
  - set.update(other, ...) – добавляет элементы из other в set

Множество

- **Неизменяемые множества.**
- **Создаются с помощью метода frozenset()**
- **Аналогичны изменяемым множествам.**
- **Нельзя применять методы, меняющие исходные множество: add(), remove(), pop(), clear() и т.д.**

Множество

- **Неизменяемые множества.**
- **Пример**

```
st_1 = {3, 7, 8, 89}
st_3 = frozenset(st_1) # неизменяемое
st_4 = {4,7, 8,12} # изменяемое
print(st_3.union(st_4)) # {3, 12, 7, 4, 8, 89}
print(st_3 | st_4)
print(st_3.pop())       # Ошибка
print(st_3.clear())     # Ошибка
```

Работа с датой и временем

Получение текущей даты и времени.

Форматирование даты и времени.

Модуль datetime.

Работа с датой и временем

- **Для работы с датой и временем в Python разработаны специальные модули datetime и calendar.**

Получение текущей даты и времени

- **Метод datetime.date.now()**

```
import datetime
print(datetime.datetime.now())
```

Результат:
2020-11-16 13:17:46.797000

Работа с датой и временем

- **Получение текущей даты и времени**
- **Метод datetime.date.today() – получение текущей даты**

```
import datetime
print(datetime.date.today())
```

Результат:
2020-11-16

- **Метод datetime.datetime.now().time() – получение текущего времени**

```
import datetime
print(datetime.datetime.now().time())
```

Результат:
13:17:46.797000

Работа с датой и временем

- **Форматирование даты и времени**
- **Для получения части даты или времени можно использовать следующие атрибуты:**
  - **year, month, day,**
  - **weekday,**
  - **hour, minute, second, microsecond**
  - **и др.**

Работа с датой и временем

- **Форматирование даты и времени**
- **Пример.**

```
import datetime
dt = datetime.date.today()
print(f"date : {dt}")
print(f"format_date: {dt.day}.{dt.month}.{dt.year}")
```

Результат:

date : 2020-11-16

format_date: 16.11.2020

Работа с датой и временем

- **Форматирование даты и времени**
- **Метод strftime() - преобразует дату в строку в соответствии с указанным форматом.**
  - **Символы формата:**
    - **%d – день месяца с 1 по 31**
    - **%m – месяц (1 – 12)**
    - **%Y, %y – год (4 или 2 знака)**
    - **%H – час в формате 0-24**
    - **%M – минуты**
    - **%S - секунды**
    - **%c – время и дата**
    - **%x – дата**
    - **%X – время**

Работа с датой и временем

- **Форматирование даты и времени**
- **Метод strftime()**

```
dt = datetime.datetime.now()
strdate = dt.strftime("%d.%m.%Y %H:%M") #строка!
print(strdate)
print(dt.strftime("%c"))
```

Результат:

16.11.2020 13:39

Mon Nov 16 13:39:37 2020

Работа с датой и временем

- **Форматирование даты и времени**
- **Метод strftime() - преобразует дату в строку в соответствии с указанным форматом.**
  - **Символы формата:**
    - **%A – полное название дня недели**
    - **%a – сокращенное название дня недели**
    - **%w – представления номера дня недели**
    - **%B – полное название месяца**
    - **%b – сокращенное название месяца**
    - **и т.д.**

Работа с датой и временем

- **Форматирование даты и времени**
- **Метод datetime.datetime.strptime(string, format) - преобразует строку в объект – дата/время в соответствии с указанным форматом.**
  - **Символы формата аналогичны методу strftime()**

```
from datetime import datetime as dtclass
date1 = datetime.datetime.strptime("13/12/1998","%d/%m/%Y")
date2 = dtclass.strptime("12-01-14 16:45","%m-%d-%y %H:%M")
print(datestr)
```

Работа с датой и временем

- **Модуль datetime**
- **Класс datetime.date(year, month, day) – создание объекта даты с аргументами (можно использовать именованные)**

```
date1 = datetime.date(1995, 9, 14)
date2= datetime.date(day=14, year=1995, month=9)
print(date1, date2)
```

Работа с датой и временем

- **Модуль datetime**
- **Класс datetime.time([hour[, minute[, second[, microsecond=0]]]]) – создание объекта времени с аргументами (можно использовать именованные)**

```
date1 = datetime.time(19, 39, 14)
date2= datetime.time(hour=19, minute=39, second=14)
print(date1, date2)
```

Работа с датой и временем

- **Модуль datetime**
- **Класс datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0) – создание объекта даты/время с аргументами (можно использовать именованные)**

```
dt1 = datetime.datetime(1995, 9, 14, 12,30)
dt2= datetime.datetime(day=14, year=1995, month=9 , minute=30, hour=12)
print(dt1, dt2)
```

Работа с датой и временем

- **Модуль datetime**
- **Методы экземпляров класса datetime.datetime:**
  - **date() – возвращает копию экземпляра даты**

```
date2= datetime.date(year=1995, month=9, day=14)
date3=date2.date()
print(date3)
```

  - **time() – возвращает копию экземпляра времени**

Работа с датой и временем

- **Модуль datetime**
- **Методы экземпляров класса datetime.datetime:**
  - **replace(year, month, …) – возвращает копию экземпляра даты/время но с замененной частью (определяется параметрами)**

```
date2= datetime.date(year=1995, month=9, day=14)
date3=date2.replace(year=2020)
print(date3)
```

Результат:
2020-09-14

Работа с датой и временем

- **Модуль datetime**
- **Методы экземпляров класса datetime.datetime:**
  - **weekday() – номер дня недели экземпляра даты (Понедельник – 0, Воскресенье - 6)**
  - **isoweekday() – номер дня недели экземпляра даты (Понедельник – 1, Воскресенье - 7)**
  - **и др.**

Работа с датой и временем

- **Продолжительность времени с timedelta**
- **datetime.timedelta – это класс, предназначенный для удобного выполнения различных манипуляций над датами и временем.**
  - ***Атрибуты класса:*** **days, hours, minutes, seconds, milliseconds, microseconds, weeks**
  - **Создание объекта timedelta:**

```
from datetime import datetime, timedelta
b = timedelta(hours=2, minutes=5, seconds=17)
```

Работа с датой и временем

- **Продолжительность времени с timedelta**
- **datetime.timedelta**
- **Операции над интервалами:**
  - **+, –, /, % - сложение, разность, деление и остаток от деления одного интервала на другой**
  - *** число – умножение каждой части интервала на указанное число**
  - **divmod(a, b)**

Работа с датой и временем

- Продолжительность времени с timedelta
- Примеры

```
from datetime import datetime, timedelta
b = timedelta(hours=2, minutes=5, seconds=17)
b1=timedelta(seconds=78)
print(f"b={b}")
print(f"b+b1={b+b1}")
print(f"b-b1={b-b1}")
print(f"b/b1={b/b1}")
print(f"b%b1={b%b1}")
print(f"divmod(b,b1)={divmod(b,b1)}")
print(f"b*2={b*2}")
```

Работа с датой и временем

- **Продолжительность времени с timedelta**
- **Примеры**

```
from datetime import datetime, timedelta
b = tim
b1=tim
print(f
print(f
print(f
print(f
print(f
print(f
print(f
print(f"b*2={b*2}")
```

Результат:

```
b=2:05:17
b+b1=2:06:35
b-b1=2:03:59
b/b1=96.37179487179488
b%b1=0:00:29
divmod(b,b1)=(96, datetime.timedelta(seconds=29))
b*2=4:10:34
```

Работа с датой и временем

- **Интервалы и даты**
- **Интервалы можно прибавлять к дате или удалять из нее**
- **Пример**

```
from datetime import datetime, timedelta
a = datetime(2020, 12, 5)
b = timedelta(hours=2, minutes=5, seconds=17)
print(f"a={a}")
print(f"b={b}")
print(f"a+b={a+b}")
print(f"a-b={a-b}")
```

Результат:

```
a=2020-12-05 00:00:00
b=2:05:17
a+b=2020-12-05 02:05:17
a-b=2020-12-04 21:54:43
```

Работа с датой и временем

- **Разность дат выражается в объекте timedelta**
- **Пример**

```
from datetime import datetime
a = datetime(2020, 12, 5)
a1 = datetime(2018, 2, 15)
b=a-a1
print(f"a={a}\na1={a1}")
print(f"b={b}")
print(f"b.days={b.days}")
print(f"b.seconds={b.seconds}")
print(f"b.total_seconds()={b.total_seconds()}")
```

Результат:

```
a=2020-12-05 00:00:00
a1=2018-02-15 00:00:00
b=1024 days, 0:00:00
b.days=1024
b.seconds=0
b.total_seconds()=88473600.0
```