Лекция «Работа со словарями. Библиотека DATE»

Что такое словарь (dict) в 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. Если вспомнить такую структуру как список, то доступ к его элементам осуществляется по индексу, который представляет собой целое неотрицательное число.

- *Создание словаря*
  Пустой словарь можно создать, используя функцию *dict()*, либо просто указав пустые фигурные скобки.

```
>>> d1 = dict()
>>> print(type(d1)) <class 'dict'>
>>> d2 = {}
>>> print(type(d2)) <class 'dict'>
```

Если необходимо создать словарь с заранее подготовленным набором данных, то можно использовать один из перечисленных выше подходов, но с перечислением групп ключ-значение.

```
>>> d1 = dict(Ivan="manager", Mark="worker")
>>> print(d1) {'Mark': 'worker', 'Ivan': 'manager'}
>>> d2 = {"A1":"123", "A2":"456"}
>>> print(d2) {'A2': '456', 'A1': '123'}
```

- *Добавление и удаление элемента*
  Чтобы добавить элемент в словарь нужно указать новый ключ и значение.

```
>>> d1 = {"Russia":"Moscow", "USA":"Washington"} >>> d1["China"]="Beijing"
>>> print(d1) {'Russia': 'Moscow', 'China': 'Beijing', 'USA': 'Washington'}
```

- Для удаления элемента из словаря можно воспользоваться командой *del*.

```
>>> d2 = {"A1":"123", "A2":"456"}
>>> del d2["A1"]
>>> print(d2) {'A2': '456'}
```

- *Работа со словарем*
  Проверка наличия ключа в словаре производится с помощью оператора *in*.

```
>>> d2 = {"A1":"123", "A2":"456"}
>>> "A1" in d2   True
>>> "A3" in d2   False
```

- Доступ к элементу словаря, осуществляется как же как доступ к элементу списка, только в качестве индекса указывается ключ.

```
>>> d1 = {"Russia":"Moscow",
"USA":"Washington"}
>>> d1["Russia"] 'Moscow'
```

Методы словарей

- **dict.clear**() - очищает словарь.
- **dict.copy**() - возвращает копию словаря.
- classmethod **dict.fromkeys**(seq[, value]) - создает словарь с ключами из seq и значением value (по умолчанию None).
- **dict.get**(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None).
- **dict.items**() - возвращает пары (ключ, значение).
- **dict.keys**() - возвращает ключи в словаре.
- **dict.pop**(key[, default]) - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение).
- **dict.popitem**() - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены.
- **dict.setdefault**(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ с значением default (по умолчанию None).
- **dict.update**([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).
- **dict.values**() - возвращает значения в словаре.

Пример задачи

Необходимо написать программу, которая будет определять к кому оператору связи относится номер телефона.

```
sv={'960':'Билайн','961':'Билайн','965':'Билайн','927':'Мегафон','937':'Мегафон'}
p=input('Номера через запятую в формате x-xxx-xxx-xx-xx')
#p='8-960-863-79-73 8-961-555-33-66 8-927-333-66-99'
p=p.split()

for i in p:
    nomer=i.split('-')[1]
    print('Данный номер телефона {} относиться к оператору {}'.format(i,sv.get(nomer)))
```

```
Номера через запятую в формате x-xxx-xxx-xx-xx8-960-863-79-73 8-961-555-33-66 8-927-333-66-99
Данный номер телефона 8-960-863-79-73 относиться к оператору Билайн
Данный номер телефона 8-961-555-33-66 относиться к оператору Билайн
Данный номер телефона 8-927-333-66-99 относиться к оператору Мегафон
```

Модуль datetime

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:

- 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).

Пример работы с классом datetime:

```
>>> from datetime import datetime, date, time
>>> # Using datetime.combine()
>>> d = date(2005, 7, 14)
>>> t = time(12, 30)
>>> datetime.combine(d, t)
datetime.datetime(2005, 7, 14, 12, 30)
>>> # Using datetime.now() or datetime.utcnow()
>>> datetime.now()
datetime.datetime(2007, 12, 6, 16, 29, 43, 79043)   # GMT +1
>>> datetime.utcnow()
datetime.datetime(2007, 12, 6, 15, 29, 43, 79060)
>>> # Using datetime.strptime()
>>> dt = datetime.strptime("21/11/06 16:30", "%d/%m/%y %H:%M")
>>> dt
datetime.datetime(2006, 11, 21, 16, 30)
>>> # Using datetime.timetuple() to get tuple of all attributes
>>> tt = dt.timetuple()
>>> for it in tt:
...     print(it)
```

```
>>> for it in tt:
...     print(it)
...
2006    # year
11      # month
21      # day
16      # hour
30      # minute
0       # second
1       # weekday (0 = Monday)
325     # number of days since 1st January
-1      # dst - method tzinfo.dst() returned None
>>> # Date in ISO format
>>> ic = dt.isocalendar()
>>> for it in ic:
...     print(it)
...
2006    # ISO year
47      # ISO week
2       # ISO weekday
>>> # Formatting datetime
>>> dt.strftime("%A, %d. %B %Y %I:%M%p")
'Tuesday, 21. November 2006 04:30PM'
```