Лекция 2
Последовательности и Словари

Давыдов Виталий Валерьевич (ГАИШ МГУ)

Повторение пройденного

- **Мультипарадигменный язык**
  - Процедурный
  - Объектно-ориентированный
  - Функциональное программирование
- **Переменные**
  - Ссылки на объекты
  - Удаление переменной (del)
- **Объекты**
  - Все в Python - объекты
  - Изменяемые (mutable) и неизменяемые (immutable)
  - Идентификатор объекта (id)
  - Тип объекта (type)
  - Тип в объекте, а не в переменной!
  - Атрибуты объекта (dir)
- **Типы**
  - Целый тип с неогранич. количеством цифр (int)
  - Числа с плавающей точкой (64 бит, float)
  - Строки (str)
  - Логический тип (bool)

Вопрос (идентификатор объекта)

- a = 2, b = 2 ⇒ id(a) и id(b) совпадают
- a = 100001, b = 100001 ⇒ id(a) ? id(b)
- b = 100001, b = 100001 ⇒ id(a) ? id(b)

Числовые Литералы и Операции

```
0
12345
-24
123456781234567812345678
1.234
1.
3.1415e-10
4E210
4.0e+210
0o123          (восьмеричный)
0x123          (шестнадцатеричный)
0b010101       (двоичный)
3.1+4.7j       (комплексный)
```

```
int(3.1415)
float(3)
bin(x), x.bit_length()
round, math.floor(3.14), math.ceil(3.14),
math.sqrt(3.14), math.sin(a)
math.pi
pow
...
```

Типы

- Встроенные типы
- Типы стандартной библиотеки Python
- Импортируемые типы (например, numpy.ndarray)
- Определенные пользователем типы
  - Аналог class в C++

Справочная система Python

- **Функция help**
  - **help**(object), где object это переменная, функция, модуль, объект класса.
  - **help**("имя"), где имя это имя модуля ключевого слова, метода, класса или раздела документации
  - **help**() без агрументов запускает интерактивную справку
  - **help**("+") - ?
- **Сайт python.org**
  - Туториал (https://docs.python.org/3/tutorial/index.html)
  - Справочник языку по Python (https://docs.python.org/3/reference/index.html)
  - Справочник по стандартной библиотеке (https://docs.python.org/3/library/index.html)
  - Предложения по улучшению PEP (https://peps.python.org/)
  - Код стайл PEP-8 (https://peps.python.org/pep-0008/)
  - Протоколы работы с объектами (https://docs.python.org/3/c-api/abstract.html)
  - Дзен Python (https://ru.wikipedia.org/wiki/Дзен_Пайтона)

Объекты

- **Объект – сущность, обладающая состоянием и поведением**
- **Python – объектно-ориентированный язык**
  - Все является объектом
  - Можно создавать новые типы объектов (классы)
  - У объекта есть атрибуты (getattr, setattr, hasattr)
- **Вопросы:**
  - Является ли тип данных объектом?
  - Является ли функция объектом?
  - Является ли импортируемый модуль объектом (например, numpy)?

Абстрактные типы объектов

- Число
- Последовательность
- Словарь
- Итератор
- Функция
- ...

Классификация встроенных типов

- **Логический тип (bool)**
- **Численные типы**
  - Целочисленный тип (int)
  - Тип с плавающей точкой (float)
- **Текстовые последовательности**
  - Юникодные строки
  - Строки
- **Последовательности**
  - Кортеж (tuple)
  - Список (list)
- **Бинарные последовательности**
  - bytes, bytearray, memoryview
- **Множества**
  - Изменяемое множество (set)
  - Неизменяемое множество (frozenset)
- **Словари**
  - dict
- **Итераторы**
  - Генераторы
- **Менеджер контекста**
- **Аннотации**
- **Типы специальных объектов**
  - None
- **Другие встроенные типы**
  - Модули, классы, функции, методы, объекта кода

Списки

- Упорядоченный массив ссылок на объекты произвольных типов
- Изменяемые (mutable, категория: sequence)
- Содержат разнородные объекты
- Доступ по индексу (смещению) элемента
- Словари не являются последовательностями

```
L = []
L = [123, 'abc', 1.23, {}]
L = ['Bob', 40.0, ['dev', 'mgr']]
L = list('spam')
L = list(range(-4, 4))
```

```
L[i] - доступ по индексу элемента
L[i][j]
len(L)
L1 + L2 – конкатенация
L * 2 – повторение
3 in L – членство
del L[2] - удаление
```

Списки / Внешние Функции

- `enumerate`
- `sorted`
- `zip`
- `sum, min, max`
- `map`
- `filter`
- `all, any`

Списки / Срезы

```
L[i] – доступ по индексу элемента
L[-1] – доступ от последнего элемента
L[1:4] – срез списка элементов с индексами 1 – 3
L[1:10:2] – срез с шагом 2
L[1:2] = [4, 5] – замена
L[1:1] = [6, 7] – вставка
L[:0] = [2, 3, 4] – вставка в начало
L[1:3] = ['A', 'B', 'C'] – присваивание по срезу
L[1:2] = [] - удаление элементов по срезу
```

Списки / Срезы / Вопросы

- `L[len(L):] = [5, 6, 7]`
- `L[len(L):0:-1]`
- `L[-len(L)]`
- `L + L`
- `L - L`
- `L ** 3 - ?`
- `L / 10`
- `L.append(L)`
- `L[0] = L`
- `L = list(range(10)), L[10] = 'A'`

Списки / Операции

list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет на i-ый элемент значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x
list.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется посл. эл.
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x
list.count(x)	Возвращает количество элементов со значением x
list.sort([key=функция])	Сортирует список на основе функции
list.reverse()	Разворачивает список
list.copy()	Поверхностная копия списка
list.clear()	Очищает список

Словари (dict)

- Неупорядоченные коллекции ссылок на произвольные объекты
- Изменяемые (mutable)
- Доступ по ключу (а не по смещению, как в списках)

```
D = {}
D = {'name': 'Bob', 'age': 40 }
D = {'cto': {'name': 'Bob', 'age': 40}}
D = dict(name='Bob", age='40')
D = dict([('name', 'Bob'),('age', 40)])
D = dict(zip(keylist, valuelist))
D = dict.fromkeys(['name', 'age'])
```

```
D['name']
D['cto']['age']
'age' in D
D.keys()
D.values()
D.items()
D.update(D2)
len(D)
D[key] = 42
del D[key]
```

Словари / Вопросы

- `D['A'] = [7, 8, 9]`
- `42 in D`
- `D.update(D)`
- `D[1] = D, del D[1][1]`

Кортежы (tuples)

- Упорядоченные коллекции произвольных объектов
- Неизменяемые (immutable, категория: sequence)
- Доступ по смещению (как списки)
- Допускают произвольное глубокое вложение
- Массивы ссылок на объекты
- Хешируемые

Кортежи

- () - пустой кортеж
- T = (0,) - одноэлементный кортеж
- T = (0, 'A', 1.2, 3) - 4-х элеменентный кортеж
- T = tuple('spam') -
- T = ('Вася', ('dev', 'mgr'))
- NT = namedtuple('Emp', ['name', 'surname'])
- a = NT(name = 'Иван', surname = 'Иванов')

Кортежи - Вопросы

- `T1 + T2`
- `T * 3`
- `T[i:j]`
- `list(('a', 'b', 'c', 'd', 'e'))`
- Могут ли быть циклы в кортеже?

Строки

- **Неизменяемые (immutable)**
- **Создание**
  - Однострочные (', ")
  - Многострочные ('''', """")
- **Доступ по индексу, нарезка**
  - s1 = s[0], s1 = s[0:1]
- **Операции**
  - Создание строки (a = "Hello")
  - Объединение строк (str.join)
- **Типы строк**
  - Юникодные строки (по-умолчанию)
  - Простые строки
  - F-строки

Строковые Литералы

```
''          (одинарные кавычки)
""          (двойные кавычки)
'Spam'
'Spa"m'
"Spam"
"Spa'm"
"s\np\ta\x00m"
'''...много.строк...'''
"""...много.строк…"""
```

```
r'c:\tmp\dir'    (неформатир)
b'sp\xc4m'       (байтовые)
u'sp\u00c4m'     (юникодные)


a = 'sp\u00c4m'
b = a.decode('unicode_escape')
```

Операция Присваивания

`a = 'Привет'`	Простое присваивание
`a, b = 'A', 'B'`	Позиц. присваивание кортежа
`[a, b] = ['A', 'B']`	Позиц. Присваивание списка
`a, b, c, d = 'Спам'`	Присваивание последовательности
`a, *b = range(10)`	Распаковка последовательности
`a = b = 'Привет'`	Групповое присваивание
`A += 42`	Дополненное присваивание: a=a+42

Операция Присваивания - Вопросы

- `a, b = b, a`
- `[a, b, c] = (1, 2, 3)`
- `[a, b] = (1, 2, 3)`
- `[a, b, c, d] = (1)`
- `s = 'Спам'; a, b, c, d = s`
- `a, b = 'Привет'[1:3]`
- `seq = [1, 2, 3, 4]; a, b, c, *d = seq`
- `a, *b, c = "Привет"`

Операция Сравнения

- `a == b`
- Сравнение поэлементное для списков и таплов