Python

Классы и ООП

basic

План

- Еще раз про наследование
- Базовый класс
- NotImplementedError
- Патерн миксин
- Перекрытие методов
- super

- чтение запись файла
- парсинг текстовых форматов

Наследование

```
class Vehicle(object):
    milage = 0

    def add_milage(self, new_milage):
        self.milage += new_milage


class Car(Vehicle):
    seating_count = 0
```

- Класс наследник будет содержать все атрибуты и методы родителя, плюс свои

Наследование, базовый класс

```
class Vehicle(object):
    milage = 0

    def add_milage(self, new_milage):
        self.milage += new_milage


class Car(Vehicle):
    seating_count = 0

class Lory(Vehicle):
    bearing_capacity = 0
```

- Общий функционал принято выносить в общий базовый класс, в классах наследниках описывать только уникальную логику

Наследование, интерфейс

```
class Vehicle(object):
    milage = 0
    def add_milage(self, new_milage):
        raise NotImplementedError


class Car(Vehicle):
    seating_count = 0
    def add_milage(self, new_milage):
        self.milage += new_milage
```

- Иногда в базовом классе удобно описать название функции и зарезать ошибку, чтоб наследники ее гарантировано перекрыли со своей логикой

Патерн миксин

```
class Vehicle(object):
    def __init__(self, brand):
        self.brand = brand

class LicenseMixin(object):
    def __init__(self, *args, **kwargs):
        self.license = kwargs.get('license')
        super(LicenseMixin, self).__init__(*args, **kwargs)

class Car(LicenseMixin, Vehicle):
    def __init__(self, brand, seat_cnt, *args, **kwargs):
        super(Car, self).__init__(brand, *args, **kwargs)
        self.seating_count = seat_cnt
```

- "Подмешивает" свой функционал не меняя функционал базового класса

Работа с файлами

```
fp = open('filename.txt', 'w')
fp.write('Hello world!')
fp.close()

fp = open('filename.txt', 'r')
fp.read()
fp.close()

fp = open('filename.txt', 'a')
fp.write('Hello world!')
fp.close()
```

- Модификаторы:

"r" - чтение

"w" - запись (файл стирается, если нету - создается)

"r+" - чтение и запись

"a" - запись в конец файла (если нету - создается)

Работа с файлами

```
fp = open('filename.txt', 'r+')
fp.write('Hello world!')
fp.seek(0)
fp.read()
fp.close()

fp = open('filename.txt', 'r')
fp.read(5)
fp.tell()
fp.read(2)
fp.tell()
fp.close()
```

- Курсор, при открытии файла есть такое понятие как курсор. Т.е. текущее место в файле от куда будет производиться чтение или запись

Работа с файлами

```
import os
size_of_file = os.path.getsize(‘test1.txt')
fp = open(‘filename.txt’, ‘r+’)
fp.seek(size_of_file)
fp.write(“Write to end of file”)
fp.seek(0)
fp.write(“Write to begin of file”)
fp.close()
```

- Курсор можно устанавливать в явном виде.
- При записи, если после курсора есть другие символы, то они будут затерты

Работа с файлами

```
fp = open('filename.txt', 'r+')
fp.read() # читает все
fp.readline() # читает одну строку
# (до символа \n)
fp.readlines() # читает все в список
# разбивая по символу перевода строк
fp.read(5) # читает количество символов
fp.close()

fp = open('filename.txt', 'r+')
fp.write("some string") # записывает строку
fp.writelines(['a', 'b', 'c'])
# записывает из итератора
fp.close()
```

CSV формат

```
fp = open(“test.csv", 'w')
csv_w = csv.writer(fp)
csv_w.writerow(['some']*5)
csv_w.writerow(['some']*5)
fp.close()
import csv
with open('test.csv', 'r') as fp:
    csv_r = csv.reader(fp)
    for row in csv_r:
        print row
```