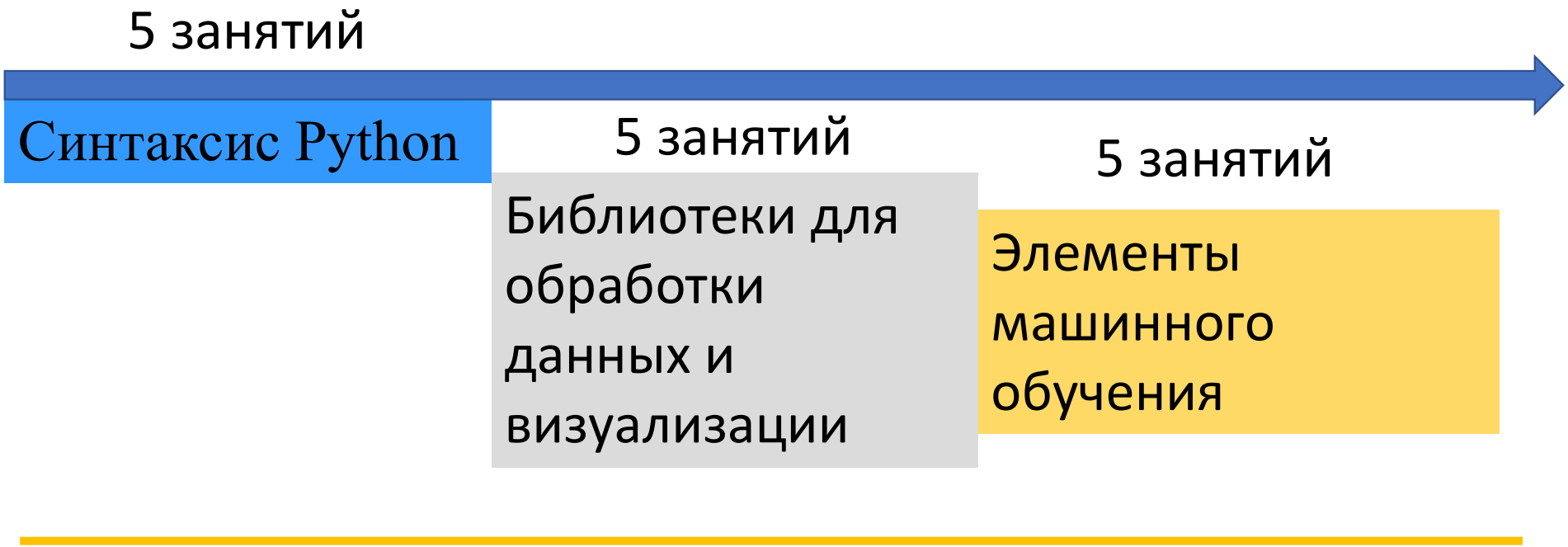
Анализ данных на Python

Медведева Светлана Юрьевна

Кафедра Информатики и вычислительной математики МФТИ

План курса


5 занятий
Синтаксис Python
5 занятий
Библиотеки для обработки данных и визуализации
5 занятий
Элементы машинного обучения

Система оценивания

- **3 ДЗ** – 10 б. *3 (одно ДЗ по каждому разделу)
- **15 тестов** – 2 б.*15 (тесты на парах начиная со второй)
- **Зачёт** – 50 б. (по списку вопросов)

Итого: 110 б. за семестр.
Оценка округляется до целого числа снизу.

Объектно-ориентированное программирование на Python.

- Понятие объекта и класса.
- Парадигмы ООП. SOLID-принципы.
- «Магические» методы классов в Python.
- Статические и классовые методы.
- Абстрактные классы.
- Декомпозиция программы на модули. Менеджер контеста.
- Обработка исключений.

Парадигма ООП

- Данные структурируются в виде объектов, каждый из которых имеет определенный тип, то есть принадлежит к какому-либо классу.
- Классы – результат формализации решаемой задачи, выделения главных ее аспектов.
- Внутри объекта инкапсулируется логика работы с относящейся к нему информацией.
- Объекты в программе взаимодействуют друг с другом, обмениваются запросами и ответами.
- При этом объекты одного типа сходным образом отвечают на одни и те же запросы.
- Объекты могут организовываться в более сложные структуры, например, включать другие объекты или наследовать от одного или нескольких объектов.

SOLID-принципы

S Принцип единственной ответственности (single responsibility principle)
Для каждого класса должно быть определено единственное назначение.

O Принцип открытости/закрытости (open–closed principle)
«программные сущности … должны быть открыты для расширения, но закрыты для модификации».

L Принцип подстановки Лисков (Liskov substitution principle)
«объекты в программе должны быть заменяемыми на экземпляры их подтипов без изменения правильности выполнения программы».

I Принцип разделения интерфейса (interface segregation principle)
«много интерфейсов, специально предназначенных для клиентов, лучше, чем один интерфейс общего назначения»

D Принцип инверсии зависимостей (dependency inversion principle)
«Зависимость на Абстракциях. Нет зависимости на что-то конкретное»

Понятие объекта и класса.

Объектно-ориентированное программирование (ООП) — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.

Класс — тип, описывающий устройство объектов. Объект — это экземпляр класса.

Простейший пример класса:

```
class C:
    pass
имя_объекта = имя_класса()
```

```
class Rectangle:
    default_color = "green"
    def __init__(self, width, height):
        self.width = width
        self.height = height
```

Статический атрибут

Динамические атрибуты

«Магические» методы классов в Python

- __new__(cls, [...) - метод, который будет вызван при инициализации объекта.
- __init__(self, [...) - инициализатор класса.
- __del__ - деструктор объекта.

```
from os.path import join

class FileObject:
    '''Обёртка для файлового объекта, чтобы быть уверенным в том, что файл будет закрыт при удалении.'''

    def __init__(self, filepath='~', filename='sample.txt'):
        # открыть файл filename в filepath в режиме чтения и записи
        self.file = open(join(filepath, filename), 'r+')

    def __del__(self):
        self.file.close()
        del self.file
```

Статические и классовые методы

```
class ToyClass:
    def instancemethod(self):
        return 'instance method called', self

    @classmethod
    def classmethod(cls):
        return 'class method called', cls


    @staticmethod
    def staticmethod():
        return 'static method called'
```

Абстрактные классы

Абстрактным называется класс, который содержит один и более абстрактных методов. **Абстрактным** называется объявленный, но не реализованный метод.

```
from abc import ABC, abstractmethod

class ChessPiece(ABC):
    # общий метод, который будут использовать все наследники этого класса
    def draw(self):
        print("Drew a chess piece")

    # абстрактный метод, который будет необходимо переопределять для каждого подкласса
    @abstractmethod
    def
        passmove(self):
```

Декомпозиция программы на модули. Менеджер контеста.

Функции модулей:

- **Повторное использование кода**: такой код может быть загружен много раз во многих местах.
- **Управление адресным пространством**: модуль — это высокоуровневая организация программ, это пакет имен, который избавляет вас от конфликтов. Каждый объект «проживает» свой цикл внутри своего модуля, поэтому модуль — это средство для группировки системных компонентов.
- **Глобализация сервисов и данных**: для реализации объекта, который используется во многих местах, достаточно написать один модуль, который будет импортирован.

Декомпозиция программы на модули. Менеджер контеста.

1. Модуль с именем my_module можно импортировать как **import my_module**.

После этого мы получаем доступ ко всем функциям определённым в модуле:

```
my_module.func1()
my_module.func2()
f1 = my_module.func1
```

2. **from my_module import func1, func2**

```
func1()
```

3. **from my_module import ***

```
func1()
```

Декомпозиция программы на модули. Менеджер контеста.

Стандартные модули:

- sys
- os
- tempfile
- fileinput
- csv
- datetime
- и многие дргуие.

Модуль sys

exit([c]) - Выход из программы. Можно передать числовой код завершения.

argv - Список аргументов командной строки.

sys.argv[0] - содержит имя запущенной программы, а остальные параметры передаются из командной строки.

platform - Платформа, на которой работает интерпретатор.

stdin, stdout, stderr - Стандартный ввод, вывод, вывод ошибок.

Version - Версия интерпретатора.

setrecursionlimit(limit) - Установка уровня максимальной вложенности рекурсивных вызовов.

exc_info() - Информация об обрабатываемом исключении.

Модуль os

os.curdir - Текущий каталог

os.pardir - Родительский каталог

os.sep - Разделитель элементов пути

os.altsep - Другой разделитель элементов пути

os.pathsep - Разделитель путей в списке путей

os.defpath - Список путей по умолчанию

os.linesep - Признак окончания строки

Модуль tempfile

В некоторых случаях необходимо создать временный файл, который после выполнения некоторых действий уже не нужен. Для этих целей можно использовать функцию **TemporaryFile**, которая возвращает файловый объект,

готовый к записи и чтению.

Модуль csv

- csv.reader
- csv.writer
- класс csv.Dictwriter - аналогичен классу DictWriter и выполняет противоположную функцию: запись данных в файл CSV.
- класс csv.DictReader - создает объект, который отображает прочитанную информацию в словарь, ключи которого задаются параметром fieldnames.

Обработка исключений

Системные исключения и ошибки

- **SystemExit**
- **KeyboardInterrupt**
- **GeneratorExit**

Обыкновенные исключения

try:
    исполняем какой-то код
except Exception as e:
    обработка исключения
else:
    код, который будет исполнен в случае, когда не возникает исключения
finally:
    код, который гарантированно будет исполнен последним (всегда исполняется)

Итоги

- Понятие объекта и класса.
- Парадигмы ООП. SOLID-принципы.
- «Магические» методы классов в Python.
- Статические и классовые методы.
- Абстрактные классы.
- Декомпозиция программы на модули. Менеджер контеста.
- Обработка исключений.

Спасибо за внимание!