

Лекция 2. Объектно-ориентированное программирование в Python

А. Н. Лата

МГИМО(У) МИД России

14 февраля 2024 г.



- 1 Парадигмы программирования
- 2 Объектно-ориентированное программирование
- 3 Объектно-ориентированное программирование в Python

Парадигмы программирования

Возникновение парадигм программирования связано с развитием языков программирования.

Парадигма программирования

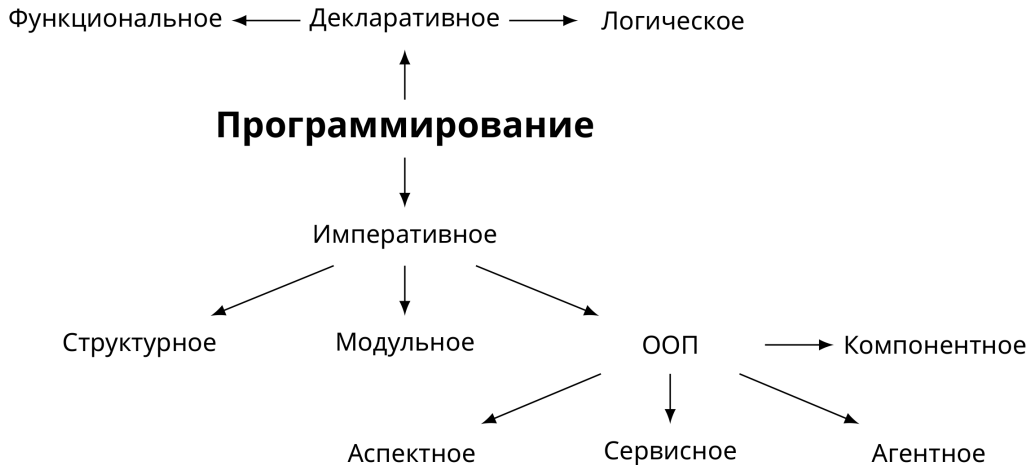
Парадигма программирования (programming paradigm) — система базовых понятий, определяющие способ написания компьютерных программ.

Парадигмы программирования затрагивают различные **аспекты** языков программирования:

- абстракции, используемые для представления данных
- подходы к декомпозиции процесса вычислений
- и т. п.

Многие языки программирования реализуют одновременно несколько парадигм программирования.

Классификация парадигм программирования

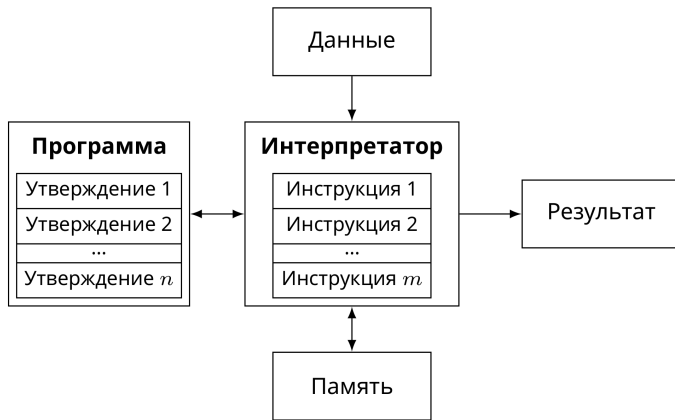


Декларативное программирование

Декларативное программирование

Декларативное программирование (англ. *declarative programming*) парадигма, согласно которой программа представляет логику вычислений без описания прямой последовательности действий (действия определяются компилятором или интерпретатором).

Выполнение декларативной программы



Декларативная программа не взаимодействует напрямую с памятью, поручая эту работу интерпретатору

Императивное программирование

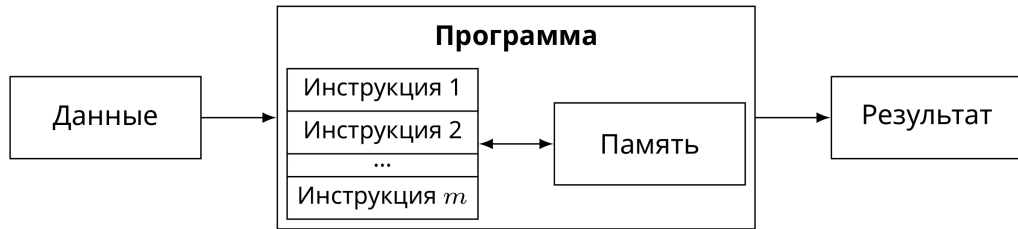
Императивное программирование

Императивное программирование (англ. *imperative programming*) парадигма, согласно которой программа представляет собой последовательность действий, изменяющих состояние программы.

Состояние программы

Состояние программы (англ. *program state*) совокупность данных, связанных со всеми используемыми программой переменными в конкретный момент времени.

Выполнение императивной программы



Императивная программа использует именованные области памяти (переменные) для хранения состояния вычислений

- 1 Парадигмы программирования
- 2 Объектно-ориентированное программирование
- 3 Объектно-ориентированное программирование в Python

ООП: классы и объекты I

ООП, object-oriented programming

Объектно-ориентированное программирование (ООП, object-oriented programming) является одной из парадигм программирования и предполагает непосредственное создание классов и объектов, а также определение связей между ними, выполняемое с использованием одного из языков объектно-ориентированного программирования.

Класс

Класс — тип данных, представляющий модель какой-то сущности.

Объект

Объект — конкретная реализация какого-то класса (**экземпляр**).

Составляющие класса (объекта)

- идентификатор;
- атрибуты (свойства);
- методы (функции).

ООП: атрибуты и методы

Атрибуты класса и атрибуты объекта

Атрибуты класса (объекта) — это переменные, характеризующие состояние или свойства класса (объекта).

Методы

Методы класса (объекта) — это функции, которые имеют непосредственный доступ к атрибутам класса (объекта). Иногда говорят, что методы определяют поведение класса (объекта).

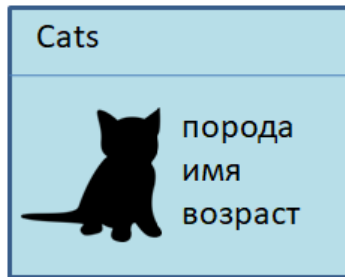
Основополагающими принципами объектно-ориентированного подхода являются

- абстракция
- инкапсуляция
- наследование
- полиморфизм

Основные принципы ООП: абстракция

Абстракция


Абстракция — придание классу объекту характеристик, которые отличают его от всех других объектов, четко определяя его концептуальные границы.



Основные принципы ООП: абстракция

Абстракция

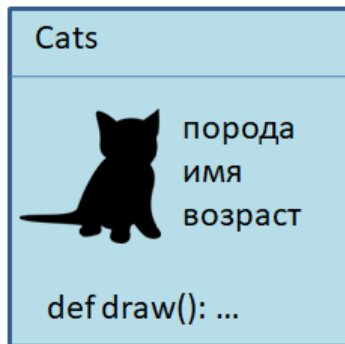
Абстракция — придание классу объекту характеристик, которые отличают его от всех других объектов, четко определяя его концептуальные границы.

Cats		
	Бурма Васька 3	
	Саванна Красик 5	
	Русская Рыжик 2	

Основные принципы ООП: абстракция

Абстракция

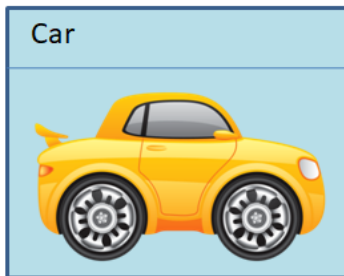
Абстракция — придание классу объекту характеристик, которые отличают его от всех других объектов, четко определяя его концептуальные границы.



Основные принципы ООП: инкапсуляция

Инкапсуляция

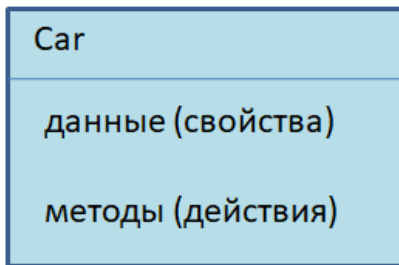
Инкапсуляция — скрывание информации от внешних (по отношению к системе или объекту) сущностей.



Основные принципы ООП: инкапсуляция

Инкапсуляция

Инкапсуляция — скрывание информации от внешних (по отношению к системе или объекту) сущностей.

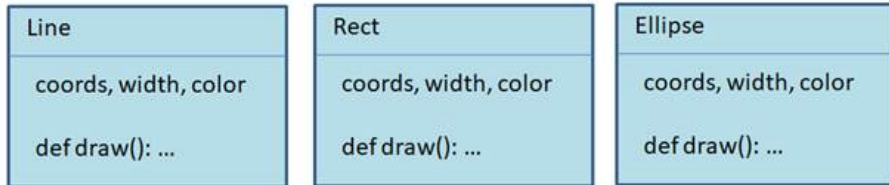


Основные принципы ООП: наследование

Наследование

Наследование — повторное использование методов работы с данными в различных условиях; дополнение функциональности объектов.

Наследование позволяет при создании нового класса указать для него **базовый класс**. От базового класса наследуется вся его структура — атрибуты и методы. Созданный класс-наследник называется **производным классом**.

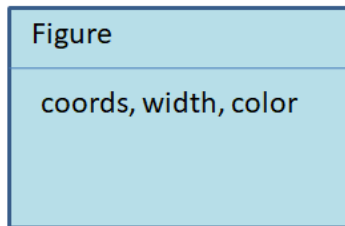


Основные принципы ООП: наследование

Наследование

Наследование — повторное использование методов работы с данными в различных условиях; дополнение функциональности объектов.

Наследование позволяет при создании нового класса указать для него **базовый класс**. От базового класса наследуется вся его структура — атрибуты и методы. Созданный класс-наследник называется **производным классом**.



Основные принципы ООП: полиморфизм

Полиморфизм

Полиморфизм — возможность использования наследованных от объекта потомков в том же контексте, что и сам объект.

Figure

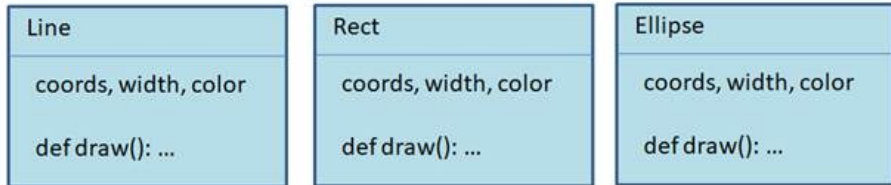
coords, width, color

def draw(): ...

Основные принципы ООП: полиморфизм

Полиморфизм

Полиморфизм — возможность использования наследованных от объекта потомков в том же контексте, что и сам объект.



ООП на основе классов

ООП на основе **классов** вид ООП, в котором поведение объектов и наследование определяется с помощью классов (начальный набор данных + поведение).

Языки программирования:

- C++;
- Java;
- C#;
- Python.

ООП на основе прототипов

ООП на основе **прототипов** вид ООП, в котором наследование поведения осуществляется с помощью клонирования существующих объектов (прототипов).

Языки программирования:

- JavaScript;
- Lua

- 1 Парадигмы программирования
- 2 Объектно-ориентированное программирование
- 3 Объектно-ориентированное программирование в Python

Язык программирования Python является объектно-ориентированным. Это означает, что каждая сущность (переменная, функция и т. д.) в этом языке является объектом определённого класса.

Пример

Класс `int` — тип данных, моделирующий целые числа.

1, 2, 15 — объекты этого класса

Убедимся в этом, написав простую программу:

```
print(type(15))
```

```
<class 'int'>
```

```
print(type(int))
```

```
<class 'type'>
```

```
print(type(type))
```

```
<class 'type'>
```

Классы и объекты в Python

Синтаксис создания класса в Python выглядит следующим образом:

```
class <ИмяКласса>:  
    <описание класса>
```

Имя класса по стандарту PEP 8 записывается в стиле **CapWords** (каждое слово с прописной буквы).

В классах описываются свойства объектов и действия объектов или совершаемые над ними действия.

Классы и объекты в Python

Свойства объектов называются **атрибутами**.

По сути атрибуты — переменные, в значениях которых хранятся свойства объекта.

Для создания или изменения значения атрибута необходимо использовать следующий синтаксис:

```
<имя_объекта>.<имя_атрибута> = <значение>
```

Чтобы увидеть **все атрибуты** класса можно обратиться к специальной коллекции:

```
<ИмяКласса>.__dict__
```

Классы и объекты в Python

Действия объектов называются **методами**.

Методы очень похожи на функции, в них можно передавать аргументы и возвращать значения с помощью оператора **return**, но вызываются методы после указания конкретного объекта.

Для создания метода используется следующий синтаксис:

```
def <имя_метода>(self, <аргументы>):  
    <тело метода>
```

В методах первым аргументом всегда идёт объект **self**. Он является объектом, для которого вызван метод. **self** позволяет использовать внутри описания класса атрибуты объекта в методах и вызывать сами методы.

Документирование кода (докстринги)

Документирование кода встроено в язык программирования Python.

Атрибут doc

У каждого объекта может быть атрибут

`__doc__`

в котором находится документация

```
int.__doc__    или    print(int.__doc__)
```

```
int([x]) -> integer
```

```
int(x, base=10) -> integer
```

```
...
```

Создание докстрингов

Строковый литерал добавляется в самом начале объявления (модуля, функции, класса).

```
def f():  
    "Документация в одну строку"  
    ...
```

```
def f(x, y, z):  
    """Документация в  
        несколько  
        строк с  
        разрывами строк"""  
    ...
```

Инициализатор и финализатор

Магические методы

В каждом классе языка Python есть набор предопределенных «магических» методов. Магические методы начинаются и заканчиваются двумя подчеркиваниями:

`__имя метода__`

В частности существуют два таких метода:

`__init__(self)` - инициализатор объекта класса
`__del__(self)` - финализатор класса