

Объектно-Ориентированное Программирование

Берленко Т.А. СПбГЭТУ “ЛЭТИ”, ФКТИ, МОЭВМ

Основные понятия. Объект

Объект - конкретная сущность предметной области.

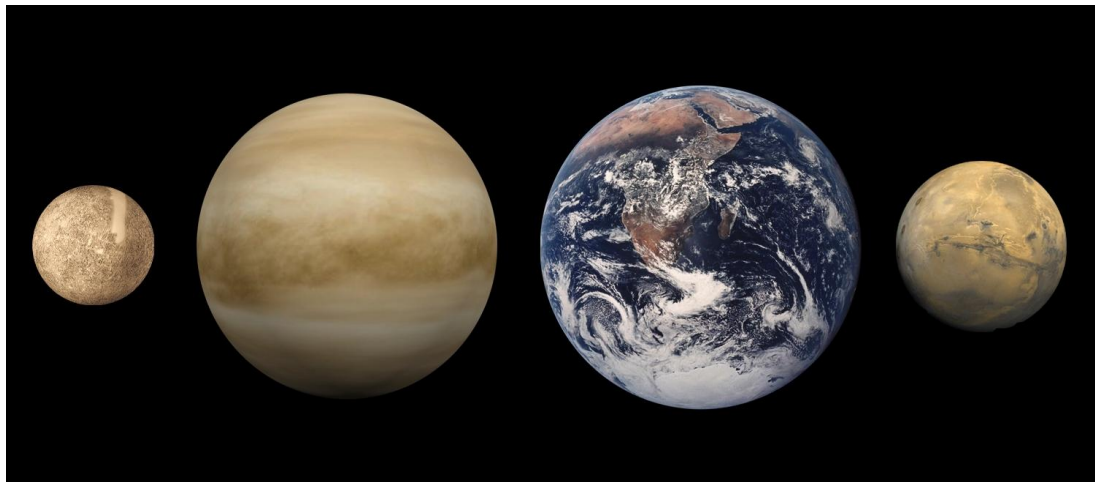


Основные понятия. Класс

Класс - это тип объекта. Или говорят, что объект - экземпляр класса.

Класс: **Планета.**

Объекты: **Меркурий, Венера, Земля, Марс.**



Пример программы “Hello, World!” на языке Java

```
class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello, World!");
    }
}
```

Основные понятия. Методы класса

- Метод класса - функция, которая принадлежит классу.
- В языке Python первый аргумент метода - `self`, экземпляр класса. При описании метода пишется явно.
- Методы имеют доступ к полям экземпляра класса.
- Пример:

```
class Student:
```

```
    def __init__(self): # метод класса Student; self - текущий экземпляр класса Student  
        pass
```

Основные понятия. Конструктор

- Конструктор - метод, который вызывается при создании экземпляра класса.
- Конструктор ничего не возвращает.
- Конструктор может быть не описан, тогда создастся пустой объект.
- Конструктор может быть только один, но может иметь переменные по умолчанию.

Основные понятия. Поля объекта

- Поле (атрибут) объекта - некоторая переменная, которая лежит в области видимости объекта и доступна во внешней программе через синтаксис:

`<имя_объекта>.<поле>`

- Поля объекта устанавливаются в *методах* класса через обращение к экземпляру `self`, например:

```
def __init__(self):
```

```
    self.name = 'Fedor' # поле экземпляра класса Student
```

Основные понятия. Создание простого класса

```
class Student:
```

```
    def __init__(self): # конструктор; self - текущий экземпляр класса Student
```

```
        self.name = 'Fedor' # поле экземпляра класса Student
```

```
        name = 'Petr' # локальная переменная в конструкторе
```

```
new_student = Student() # создание экземпляра класса Student
```

```
print(new_student.name)
```


Некоторые основные принципы ООП

- Инкапсуляция
- Наследование
- Полиморфизм

Инкапсуляция

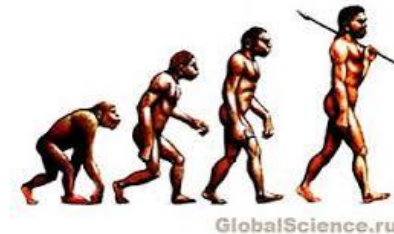
- Соккрытие внутренней реализации от пользователя.

! В языке Python нельзя создать по-настоящему приватный атрибут.

- Соккрытие деталей реализации за интерфейсом объекта.

Пример: работа с библиотекой Wikipedia.

Наследование



- Повторное использование и последующее расширение одним классом атрибутов другого класса.
- Класс, определённый через наследование от другого класса, называется производным классом, классом-потомком, подклассом, классом-наследником. Класс, от которого новый класс наследуется, называется предком, базовым классом или суперклассом.
- Пример в Python:

```
class Super: # Родитель
```

```
    def __init__(self):  
        self.data = 'some data'
```

```
class Subclass(Super): # Потомок
```

```
    pass
```

Класс object

- Встроенный класс object является суперклассом для всех классов.
- В иерархии наследования класс object автоматически стоит выше остальных классов.
- Класс object предоставляет методы, которые реализуют различные операции, например, поиск длины объекта или вывод на экран. Эти методы можно переопределить.

isinstance() и isinstance()

➤ `isinstance(obj1, class1)`

Возвращает True, если **obj1** является экземпляром класса **class1** или суперкласса класса **class1**

➤ `issubclass(class1, class2)`

Возвращает True, если **class1** является наследником класса **class2**.

super()

- Иногда в процессе написания метода в классе-наследнике может понадобиться вызвать метод суперкласса. Это можно сделать через имя суперкласса или через функцию `super()`.

- **class** `C(B) :`
`def method(self, arg) :`
`super().method(arg) # То же самое, что:`
`# super(C, self).method(arg)`

Вызов конструктора суперкласса

```
class A:  
    def __init__(self, name):  
        self.name = name  
  
class B(A):  
    def __init__(self, name, age):  
        super().__init__(name)  
        self.age = age
```

```
b = B('Olya', 10)  
  
print(b.name, b.age)
```

Полиморфизм

- Способность функции обрабатывать разные типы данных, если эти данные могут поддерживать соответствующий интерфейс.
- Возможность обрабатывать объекты разных типов одинаковым образом, не задумываясь о типе каждого объекта.

Перегрузка операторов

- В Python существует возможность переопределения не только методов класса, но и операторов выражений.
- Вы можете создать свой тип данных и определить для его экземпляров операции сложения/сравнения/извлечения среза и т.д.

Поля класса

```
class Counter:  
    counter = 0 # Неизменяемое поле класса
```

`Counter.counter` *# Доступ к полю класса без создания объекта*

`Counter().counter` *# Доступ к полю класса через объект*

Поля класса

Объект может присвоить полю другое значение:

```
class Counter:  
    counter = 0 # Неизменяемое поле класса
```

```
a = Counter()  
print(a.counter, Counter.counter) # 0 0  
a.counter = 10  
print(a.counter, Counter.counter) # 10 0
```

Поля класса

```
class Student:  
    marks = [] # Изменяемое поле класса
```

`Student.marks` *# Доступ к полю класса без создания объекта*

`Student().marks` *# Доступ к полю класса через объект*

Поля класса

Объект может изменить поле класса:

```
class Student:  
    marks = [] # Изменяемое поле класса  
  
Alex = Student()  
print(Alex.marks, Student.marks) # [] []  
Alex.append(5)  
print(Alex.marks, Student.marks) # [5] [5]
```

Поля класса

Объект может изменить поле класса:

```
class Student:  
    marks = [] # Изменяемое поле класса
```

```
Alex = Student()
```

```
Mary = Student()
```

```
print(Alex.marks, Mary.marks, Student.marks) # [] [] []
```

```
Alex.marks = []
```

```
Mary.marks.append(5)
```

```
print(Alex.marks, Mary.marks, Student.marks) # [] [5] [5]
```

Поля класса при наследовании

```
class A:
```

```
    field = 10
```

```
class B(A):
```

```
    pass
```

```
print(B.field) # 10
```

```
print(B().field) # 10
```

Источники и полезные ссылки

- Классы в Python, документация

<https://docs.python.org/3/tutorial/classes.html>

- Операторы языка python, которые вы можете переопределить:

<https://docs.python.org/3/reference/datamodel.html>

- Курс “Python: основы и применение” <https://stepik.org/course/512/syllabus>

- Лутц “Изучаем Python”