ЯрГУ им. П.Г. Демидова

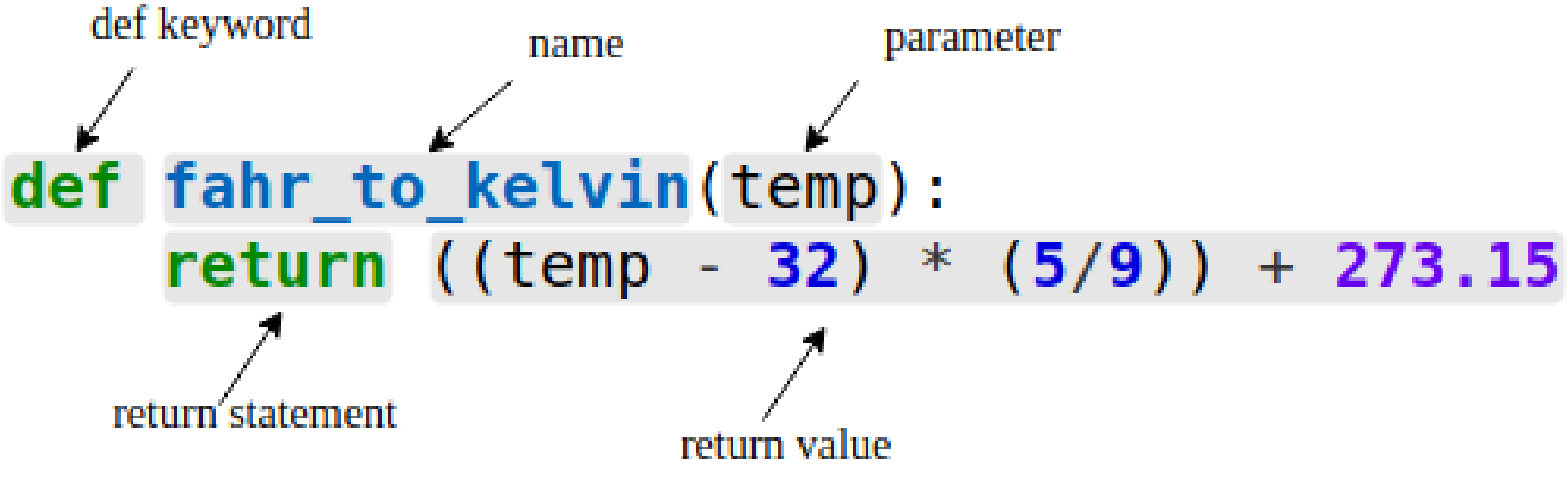
ЭЛЕМЕНТЫ ООП в Python

Андрей Рагимов
andrey@ragimov.info

ФУНКЦИИ

СИНТАКСИС

Пример объявления функции с одним аргументом



ФУНКЦИИ

ЛЯМБДА-ФУНКЦИИ

```
>>> fib = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
>>> result = filter(lambda x: x < 7, fib)
>>> print(list(result))
[0, 1, 1, 2, 3, 5]
```

KEYWORD АРГУМЕНТЫ

```
>>> def func(param1, param2):
...     print("param1 is {param1}, param2 is
{param2}".format(param1=param1, param2=param2))
...
>>> func(5, param2=10)
param1 is 5, param2 is 10
```

ФУНКЦИИ

ОСОБЕННОСТИ ЗАДАНИЯ ЗНАЧЕНИЙ ПО УМОЛЧАНИЮ

Изменяемые типы не стоит задавать как значения по умолчанию

```
>>> def wrong_function(x=5, some_list=[]):
...     some_list.append(x)
...     return some_list
...
>>> wrong_function()
[5]
>>> wrong_function()
[5, 5]
```

ФУНКЦИИ

ОСОБЕННОСТИ ЗАДАНИЯ ЗНАЧЕНИЙ ПО УМОЛЧАНИЮ

В таких случаях используйте None:

```
>>> def func(x=5, some_list=None):
...     if some_list is None:
...         some_list = []
...     some_list.append(x)
...     return some_list
...
>>> func()
[5]
>>> func()
[5]
```

ЗАДАЧА

ВЫЧИСЛИТЕ ДВАДЦАТОЕ ЧИСЛО ФИББОНАЧИ

Числа Фибона́ччи — элементы числовой последовательности
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, …
в которой первые два числа равны либо 1 и 1, либо 0 и 1, а каждое последующее число равно сумме двух предыдущих чисел.

$$F_0 = 0, \qquad F_1 = 1, \qquad F_n = F_{n-1} + F_{n-2}, \quad n \geqslant 2, \quad n \in \mathbb{Z}.$$

РЕШЕНИЕ

ПРИМЕР РЕШЕНИЯ

Простая, но неэффективная реализация

```
>>> def fibonacci(n):
...     if n == 0:
...         return 0
...     elif n == 1:
...         return 1
...     else:
...         return fibonacci(n-1) + fibonacci(n-2)
...
>>> fibonacci(20)
6765
```

ООП

КЛАСС И ЭКЗЕМПЛЯР КЛАССА

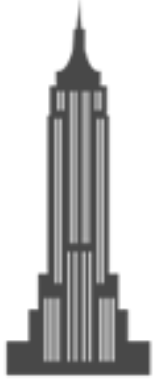
Class
Definition of objects that share structure, properties and behaviours.

Building
class

Dog
class

Computer
class

Instance
Concrete object, created from a certain class.



Empire State
instance of Building

Lassie
instance of Dog

Your computer
instance of Computer

ООП

НАСЛЕДОВАНИЕ

Наследование – это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым или родительским. Новый класс – потомком, наследником или производным классом.

Задание: приведите примеры

ООП

ПОЛИМОРФИЗМ

Полиморфизм – это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.
Задание: приведите примеры.

ИНКАПСУЛЯЦИЯ

Инкапсуляция – это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали
реализации от пользователя.
Задание: приведите примеры

ООП

КЛАСС И ЭКЗЕМПЛЯР КЛАССА В PYTHON

```
>>> class Parrot:
...     def __init__(self, name):
...         self.name = name
...
...     def say_name(self):
...         print("Меня зовут %s" % self.name)
...
>>> kesha = Parrot("Кеша")
>>> kesha.say_name()
Меня зовут Кеша
```

ООП

НАСЛЕДОВАНИЕ В PYTHON

```
>>> class Ship:
...     def swim(self):
...         print("Плывем по волнам")
...
>>> class MilitaryShip(Ship):
...     def open_fire(self):
...         print("Открыть огонь!")
...
>>> varyag = MilitaryShip()
>>> varyag.swim()
Плывем по волнам
>>> varyag.open_fire()
Открыть огонь!
```

ООП

ПОЛИМОРФИЗМ В PYTHON

```
>>> class Auto:
...     def drive_using_wheel(self):
...         print("Управляем автомобилем с помощью руля")
...
>>> class Truck:
...     def drive_using_wheel(self):
...         print("Управляем грузовиком с помощью руля")
...
>>> for vehicle in [Auto(), Truck()]:
...     vehicle.drive_using_wheel()
...
Управляем автомобилем с помощью руля
Управляем грузовиком с помощью руля
```

ЗАМЕНА ИНКАПСУЛЯЦИИ В PYTHON

```
>>> class TV:
...     def __init__(self):
...         self.diode = "Красный"
...         self._status = "Выключен"
...
...     def _initialize(self):
...         print("Инизиализируем включение телевизора")
...
...     def turn_on(self):
...         self.diode = "Зеленый"
...         self._status = "Инициализация"
...         self._initialize()
...         self._status = "Включен"
```

ООП

ЗАМЕНА ИНКАПСУЛЯЦИИ В PYTHON

```
>>> tv = TV()
>>> print(tv.diode)
Красный
>>> tv.turn_on()
Инизиализируем включение телевизора
>>> print(tv.diode)
Зеленый
>>> print(tv._status)  # Есть доступ к "приватному" полю
Включен
```

В языке Python отсутствует инкапсуляция как таковая, но вместо этого используется принятое соглашение именовать приватные (скрытые) переменные с нижнего подчеркивания.

ЗАДАЧА

СОЗДАЙТЕ ИЕРАРХИЮ КЛАССОВ