UPPSALA UNIVERSITET

MA3 introduction

Tom Smedsaas

This module is about *data structures*, mostly *linked lists* and *trees*.

We will also discuss some new features:

- Operator overloading
- Iterators
- Generators

However this short lecture is just a repetition of what classes and objects are.

UPPSALA UNIVERSITET

Everything in Python is an object

Examples:

- Numbers
- Strings
- Lists
- Dictionaries
- Functions
- Modules
- and more …

Exceptions: `+`, `<`, `in`, `not`, `and`, `for`, `if`, `[`, `)`, …

UPPSALA UNIVERSITET

Objects have properties

The function `dir` can be used to see the properties.
Example:

```
>>> dir(4.6)
['__abs__', '__add__', '__bool__', '__class__', '__delattr__', '__dir__', '__divmod__',
'__doc__', '__eq__', '__float__', '__floordiv__', '__format__', '__ge__', '__getattribute__',
'__getformat__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__int__', '__le__', '__lt__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__',
'__pos__', '__pow__', '__radd__', '__rdivmod__', '__reduce__', '__reduce_ex__',
'__repr__', '__rfloordiv__', '__rmod__', '__rmul__', '__round__', '__rpow__', '__rsub__',
'__rtruediv__', '__set_format__', '__setattr__', '__sizeof__', '__str__', '__sub__',
'__subclasshook__', '__truediv__', '__trunc__', 'as_integer_ratio', 'conjugate', 'fromhex',
'hex', 'imag', 'is_integer', 'real']
>>> 4.6.__round__()
5
>>> round(4.6)
5
```

UPPSALA UNIVERSITET

Python list properties

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',
'__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__',
'__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index',
'insert', 'pop', 'remove', 'reverse', 'sort']
```

Here we can see methods that we actually use, e.g. `append`, `reverse` and `sort`.

We can also see methods defining some list operators e.g. `__add__` for +, `__eq__` for == and `__le__` for <=.

These are called *dunder, magic* or *special methods.*

How create new types of objects?

Write a class!

Example from the traffic simulation in Prog 1:

```
class Vehicle:
    def __init__(self, destination, borntime):
        self.destination = destination
        self.borntime = borntime

    def __str__(self):
        return f'Vehicle({self.destination}, {self.borntime})'
```

Class name

Initializer or *constructor*

String representation

Usage:

```
car1 = Vehicle('Uppsala', 25)
car2 = Vehicle('Stockholm', 47)
print(car1)
print(car2)
print(car1.destination)
print(car2.destination)
print(car2.borntime)
```

Output:

```
Vehicle(Uppsala, 25)
Vehicle(Stockholm, 47)
Uppsala
Stockholm
47
```

UPPSALA UNIVERSITET

Another example from the traffic simulation

```
class Light:

    def __init__(self, period, green_period):
        self._period = period
        self._green_period = green_period
        self._time = 0

    def is_green(self):
        return self._time < self._green_period

    def __str__(self):
        if self.is_green():
            return "(G)"
        else:
            return "(R)"

    def step(self):
        self._time = (self._time+1) % self._period
```

The constructor.
Three *instance variables*

Predicate: `True` or `False`

Special method for converting to string

Time stepping

UPPSALA UNIVERSITET

Usage of the `Light` class

Code:

```
s1 = Light(5,2)
s2 = Light(7,3)
s3 = s2
for i in range(8):
    print(i, s1, s2, s3)
    s1.step()
    s2.step()
```

Output:

```
0 (G) (G) (G)
1 (G) (G) (G)
2 (R) (G) (G)
3 (R) (R) (R)
4 (R) (R) (R)
5 (G) (R) (R)
6 (G) (R) (R)
7 (R) (G) (G)
```

Link to the traffic simulation lesson in English and in Swedish

UPPSALA UNIVERSITET

Another example

```
class Person:
    def __init__(self, name):
        self.name = name
        self.children = []

    def __str__(self):
        return f"{self.name} : {str([s.name for s in self.children])}"

    def add_child(self, child):
        self.children.append(child)
```

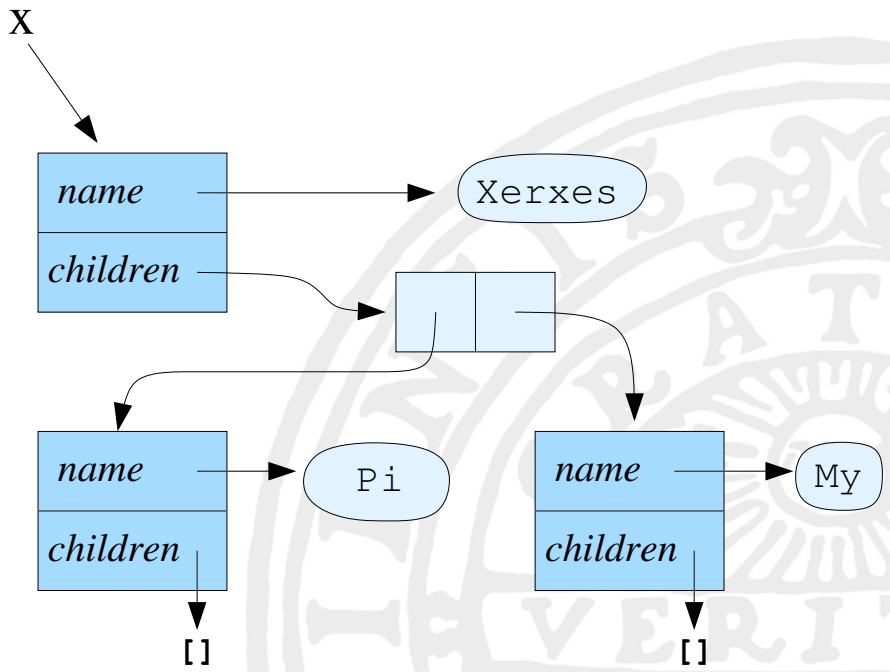
Usage of the class Person

Code:

```
x = Person("Xerxes")
x.add_child(Person("Pi"))
x.add_child(Person("My"))
print(x)
```

Output:

```
Xerxes : ['Pi', 'My']
```



UPPSALA UNIVERSITET

Summary

- Classes are used to define new types of objects
- A class contains methods and data attributes
- The `__init__` method is used to initilize an object
- The method definitions must have `self` as the first parameter
- The method `__str__` is used to define a string representation of the object
- Other special methods can define other operations on the objects (`__lt__`, `__eq__`, ...)

You will see more examples in the coming material.

The end

UPPSALA
UNIVERSITET