

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

User Defined Types

Lect Phd. Arthur Molnar

Babes-Bolyai University

Overview

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods.
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

- 1 User defined types
 - Why define new types?
 - Classes
 - Objects
 - Methods, Fields
 - Special methods. Overloading
- 2 Python scope and namespace
 - Class vs instance attributes
- 3 Principles when defining new data types
- 4 First Test

User defined types

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?

Classes

Objects

Methods, Fields
Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

NB!

Types classify values. A type denotes a **domain** (a set of values) and **operations** on those values.

User defined types

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

- **Object oriented programming** - a programming paradigm that uses objects that have data and which "talk" to each other to design applications.

Why define new types?

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

Let's review the modular calculator example:

1 Issues with global variables, if they exist:

- You can easily break global vars!
- They make testing difficult
- Managing the relation between them is difficult

2 Issues without global variables:

- The state of the calculator is exposed to the world
- The state has to be transmitted as parameter to every function

User defined types - classes

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?

Classes

Objects

Methods, Fields
Special methods,
Overloading

Python scope and namespace

Class vs instance
attributes

Principles when defining new data types

First Test

Class - a construct used as a template to create instances of itself - referred to as class instances, class objects, instance objects or simply **objects**. A class defines constituent members which enable these class instances to have *state* and behaviour.

Classes in Python

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields
Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

- Defined using the keyword **class** (as in many other languages)
- The class definition is an executable statement.
- The statements inside a class definition are usually function definitions, but other statements are allowed
- When a class definition is entered, a new namespace is created, and used as the local scope - thus, all assignments to local variables go into this new namespace. In particular, function definitions bind the name of the new function here.

User defined types - objects

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

Object - in object-oriented programming, an object refers to a particular instance of a class, and is a combination of variables, functions and other data structures. Objects support two kinds of operations: **attribute (data or method) references** and **instantiation**.

User defined types - objects

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope and namespace

Class vs instance
attributes

Principles when defining new data types

First Test

1 Object instantiation - uses the reserved function notation of **`__init__`**

- The instantiation operation creates an empty object that is of the type of the given class
- A class may define a special method named **`__init__`**, used to create an instance of that class (e.g. class → object)
- In Python, use **`self`** to refer to that instance (in many other languages, it is the **`this`** keyword)

User defined types - objects

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?

Classes

Objects

Methods, Fields
Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

2 Attribute references (method or field)

- Uses the "dot-notation", not dissimilar to package.module names.
- We have instance variables/methods and class variables/methods
- Instance variables are specific to an object (each object has its own instance)
- Class variables are specific to a class (they are shared by all instances of that class)
- The variable referencing the object specifies on which instance the call is made, in the case of instance variables

Existing data types

ex11_existingDataTypes.py

Fields, Methods

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

■ Fields

- Variables that store data specific to an instance or a class (see the slide above)
- Can be objects themselves
- They come into existence first time they are assigned to

■ Methods

- Functions in a class that can access values from a specific instance.
- In Python the method will automatically receive a first argument: the current instance
- All instance methods need to have the **self** argument

Fields, Methods

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

Demo

A first example using classes in Python -
ex12_pythonClassParticularities.py

Demo

Let's create a new data type - RationalNumber. (Source code
is in **ex13_rationalNumberBasic.py**)

Special methods

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

- **`--str--`** - converts the current object into a string type (good for printing)
- **`--eq--`** - test (logical) equality of two objects
- **`--ne--`** - test (logical) inequality of two objects
- **`--lt--`** - test $x < y$
- Many others at¹

¹<https://docs.python.org/3/reference/datamodel.html>

Special methods - operator overloading

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods.
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

- **`--add__(self, other)`** - to be able to use `" + "` operator
- **`--mul__(self, other)`** - to be able to use the `" * "` operator
- **`--setitem__(self, index, value)`** - to make a class behave like an array/dictionary, use the `" [] "`
- **`--getitem__(self, index)`** - to make a class behave like an array
- **`--len__(self)`** - overload `len`
- **`--getslice__(self, low, high)`** - overload slicing operator
- **`--call__(self, arg)`** - to make a class behave like a function, use the `" () "`

Special methods - example

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods.
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

Demo

We should make our rational number type a bit more useful.
(Source code is in **ex14_rationalNumberOperators.py**)

Python scope and namespace

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?
Classes
Objects
Methods, Fields
Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

NB!

- A *namespace* is a mapping from names to objects.
- Namespaces are implemented as Python dictionaries
 - Key: name
 - Value - Object
- Remember **globals()** and **locals()** ?

Python scope and namespace

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?
Classes
Objects
Methods, Fields
Special methods,
Overloading

Python scope and namespace

Class vs instance
attributes

Principles when defining new data types

First Test

- A class introduces a new namespace
- Methods and fields of a class are in a separate namespace (the namespace of the class)
- All the rules (bound a name, scope/visibility, formal/actual parameters, etc.) related to the names (function, variable) are the same for class attributes (methods, fields). Keep in mind that the class has its own namespace

Class vs instance attributes

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

■ Instance attributes

- The **self** reference decides for what object the attribute is accessed
- Each instance has its own set of fields

■ Class attributes

- Attributes that are unique to the class
- They are shared by all instances of the same class
- In most languages, they are referred to as "static" fields, or methods
- In Python, the **@staticmethod** decorator is used
- Static methods do not receive the **self** reference

Class vs instance attributes

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields
Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

Demo

ex15_instanceVsClassAttributes.py

Class vs instance attributes

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields
Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

Discussion

Can you think of examples where class attributes are more suitable rather than instance attributes?

Encapsulation

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope and namespace

Class vs instance
attributes

Principles when defining new data types

First Test

- A set of rules or guidelines that you will use when deciding on the implementation of new data types
- What we will cover
 - Encapsulation
 - Information hiding
 - Abstract data types

Encapsulation

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope and namespace

Class vs instance
attributes

Principles when defining new data types

First Test

- The **state** of the object is the data that represents it (in most cases, the class fields)
- The **behaviour** is represented by the class methods
- Encapsulation means that **state** and **behaviour** are kept together, in one **cohesive** unit

Information hiding

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields
Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

- The internal representation of an object needs to be hidden from view outside of the object's definition
- Hiding the internals of the object protects its integrity by preventing users from setting the internal data of the component into an invalid or inconsistent state
- Divide the code into a public interface, and a private implementation of that interface

Information hiding

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

- Defining a specific interface and isolate the internals to keep other modules from doing anything incorrect to your data
- Limit the functions that are visible (part of the interface), so you are free to change the internal data without breaking the client code
- Write to the Interface, not the the Implementation
- If you are using only the public functions you can change large parts of your classes without affecting the rest of the program

Information hiding

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?
Classes
Objects
Methods, Fields
Special methods,
Overloading

Python scope and namespace

Class vs instance
attributes

Principles when defining new data types

First Test

Public and private members - data hiding in Python

- We need to protect (hide) the internal representation (the implementation)
- Provide accessors (getter) to the data
- Encapsulations is particularly important when the class is used by others

Information hiding

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?
Classes
Objects
Methods, Fields
Special methods,
Overloading

Python scope
and
namespace
Class vs instance
attributes

Principles
when defining
new data
types

First Test

Public and private members - data hiding in Python

- Nothing in Python makes it possible to enforce data hiding - it is all based upon convention. use the convention: `_name` or `__name` for fields, methods that are "private"
- A name prefixed with an underscore (e.g. `_spam`) should be treated as a non-public part of the API (whether it is a function, a method or a data member). It should be considered an implementation detail and subject to change without notice.
- A name prefixed with two underscores (e.g. `__spam`) is private and name mangling (its actual name is replaced by the Python runtime) is employed

Guidelines

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

- Upper application layers do not have to know about implementation details of the methods or the internal data representation used by the code they call
- Code must work even when the implementation or data representation are changed
- Function and class specification have to be independent of the data representation and the method's implementation
(Data Abstraction)

Abstract data types

Lecture 06

Lect Phd.
Arthur Molnar

User defined types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope and namespace

Class vs instance
attributes

Principles when defining new data types

First Test

- Operations are specified independently of their implementation
- Operations are specified independently of the data representation
- Abstract data type is a Data type + Data Abstraction + Data Encapsulation

Week 7 Test

Lecture 06

Lect Phd.
Arthur Molnar

User defined
types

Why define new
types?

Classes

Objects

Methods, Fields

Special methods,
Overloading

Python scope
and
namespace

Class vs instance
attributes

Principles
when defining
new data
types

First Test

- First test will be during week 7's lecture (dry run during week 6 lecture)
- You will be given a problem statement to solve in 80 minutes
- Test is open book, but must be taken individually
- Solutions will be checked for plagiarism
- Use modular programming, functions, but not classes
- Weight is 20% of laboratory grade (around the same as the first 5 lab assignments)