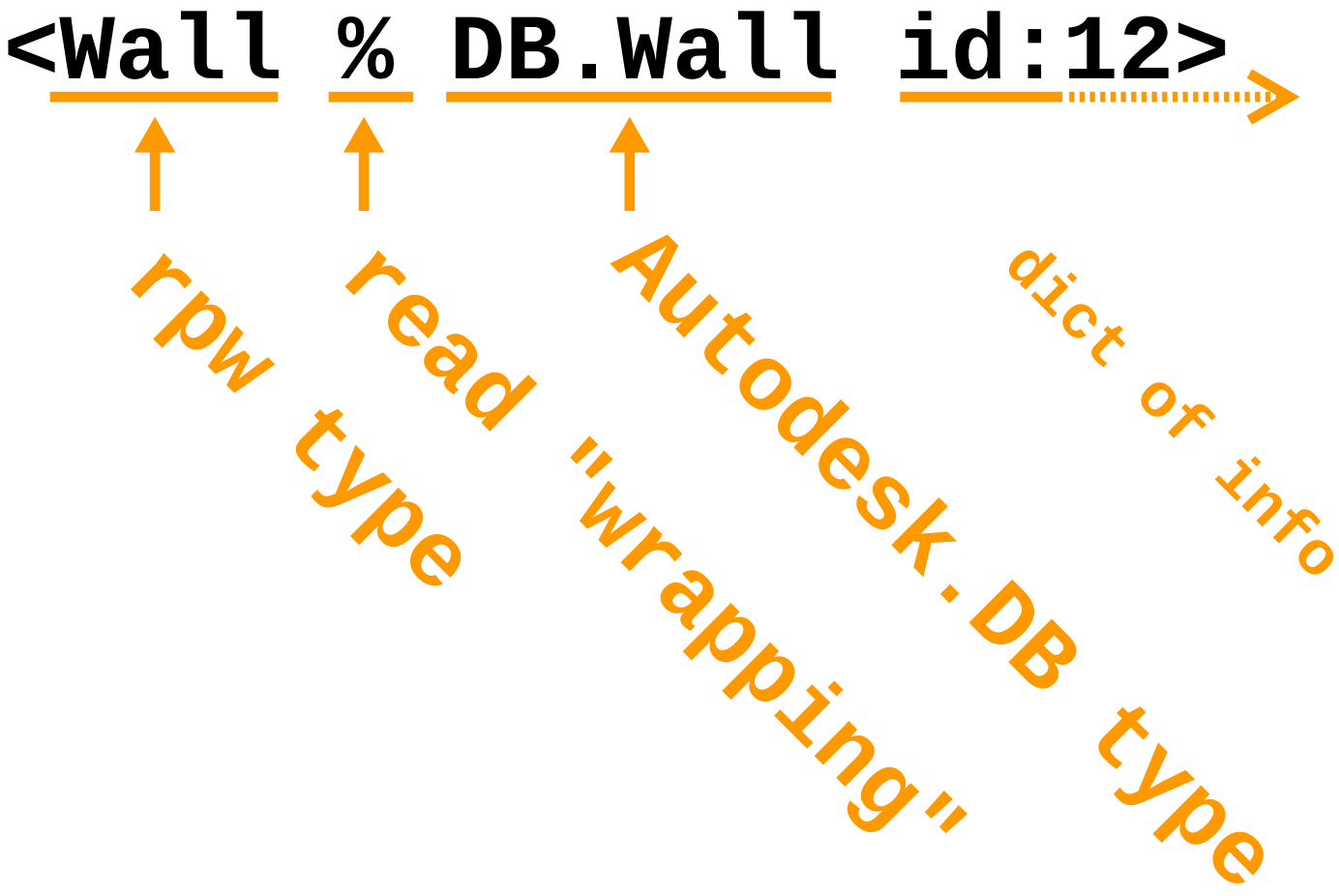
rpw

reloaded

proof of concept


rpw
IMPORTS SUB-MODULES - EXPOSES A CLEAN API
rpw
app
PROVIDES ACCESS TO HOST APP INFO, API, USER INTERFACE, DOCS
a
__init__.py
host app information (pid, screen, ...)
api.py
api handles (DB, UI, ...)
doc.py
doc, uidoc, all_docs
forms.py
Forms sub_module: provide access to host app interface and generic forms
userinput.py
base
MOST BASIC WRAPPERS FOR RPW AND DOTNET API
b
__init__.py
BaseObject, BaseWrapperObject, BaseEnumWrapper
coreutils
BASIC UTILITIES FOR SUB-MODULES
c
__init__.py
config.py
RPW config (e.g. rpw.config.wrap_all = True #wraps all elements)
logger.py
logging system based on pyRevit
db
DATABASE ACCESS SERVICES
d
core
BASIC DB WRAPPERS
__init__.py
builtins.py
BuiltInParameter & BuiltInCategory wrappers
collector.py
FilteredCollector wrappers
datamodel.py
Instance / Symbol / Family / Category data model (Element subclasses)
element.py
basic Element wrapper (any API object that has Id)
Also includes ElementCollection for lists of elements
geometry.py
parameter.py
Each element includes set of parameters (and possibly geometry)
transaction.py
Transaction, TransactionGroup wrappers
env
DB ENVIRONMENT (PROJECT, SELECTION, ...)
__init__.py
project.py
Loose wrapper for ProjectInfo
selection.py
Access to current selection and element picking methods
extensions
EXTENDED DB WRAPPERS (WALL, AREA, ...)
provides a chance to mask-out api inconsistencies
__init__.py
ext_categories.py
ext_families.py
ext_instances.py
ext_symbols.py
Extended individual wrappers
subclasses of data model types
geometric.py
Extended types for geometry classes
spacial.py
Extended types for spacial elements (Room, Area, Space)
wall.py
Extended data model types for Wall (WallInstance, WallSymbol, ...)
__init__.py
autocategories.py
Auto collector types e.g. rpw.Walls.instances()
coerce.py
Input cleanup utilities
__init__.py

repr


<Wall % DB.Wall id:12>
rpw type
read "wrapping"
Autodesk.DB type
dict of info

rpw.Element

```
class Element(BaseObjectWrapper):
    """Wrapper for all API Objec
    """
    _wrapped_class = DB.Element

    def __new__(cls, api_obj, en

    def __init__(self, api_obj,

    def __dir__(self):

    def __getattr__(self, attr):

    def __setattr__(self, attr, value):

    @property
    def id(self):

    def __repr__(self,
```

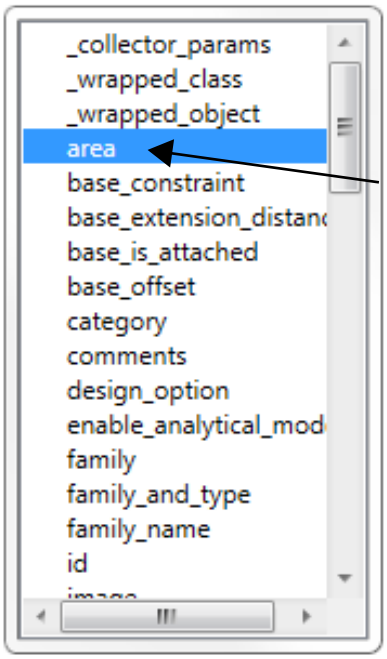
__new__ replaces all other Factory methods.

Finds subclasses and creates relevant wrapper type

```
>>>
... import rpw
... a = rpw.Element(el.Category)
... b = rpw.Element(BuiltInCategory.OST_Walls)
... c = rpw.Element(el)
... d = rpw.Element(el.Id)
... print a
... print b
... print c
... print d
...

<WallCategory % DB.Category id:-2000011>
<WallCategory % DB.Category id:-2000011>
<Wall % DB.Wall id:2360 symbol_name:Wall 1>
<Wall % DB.Wall id:2360 symbol_name:Wall 1>
```

provide clean dir for autocompletion.
provide access to parameters as attributes

```
>>> import rpw
>>> wall = rpw.Element(el)
>>> wall
<Wall % DB.Wall id:2360 symbol_name:Wall 1>
>>> wall.
```



Normally:
Element.parameters['Area']

rpw.Collector

Extended filters. Filters can change behaviour based on value.

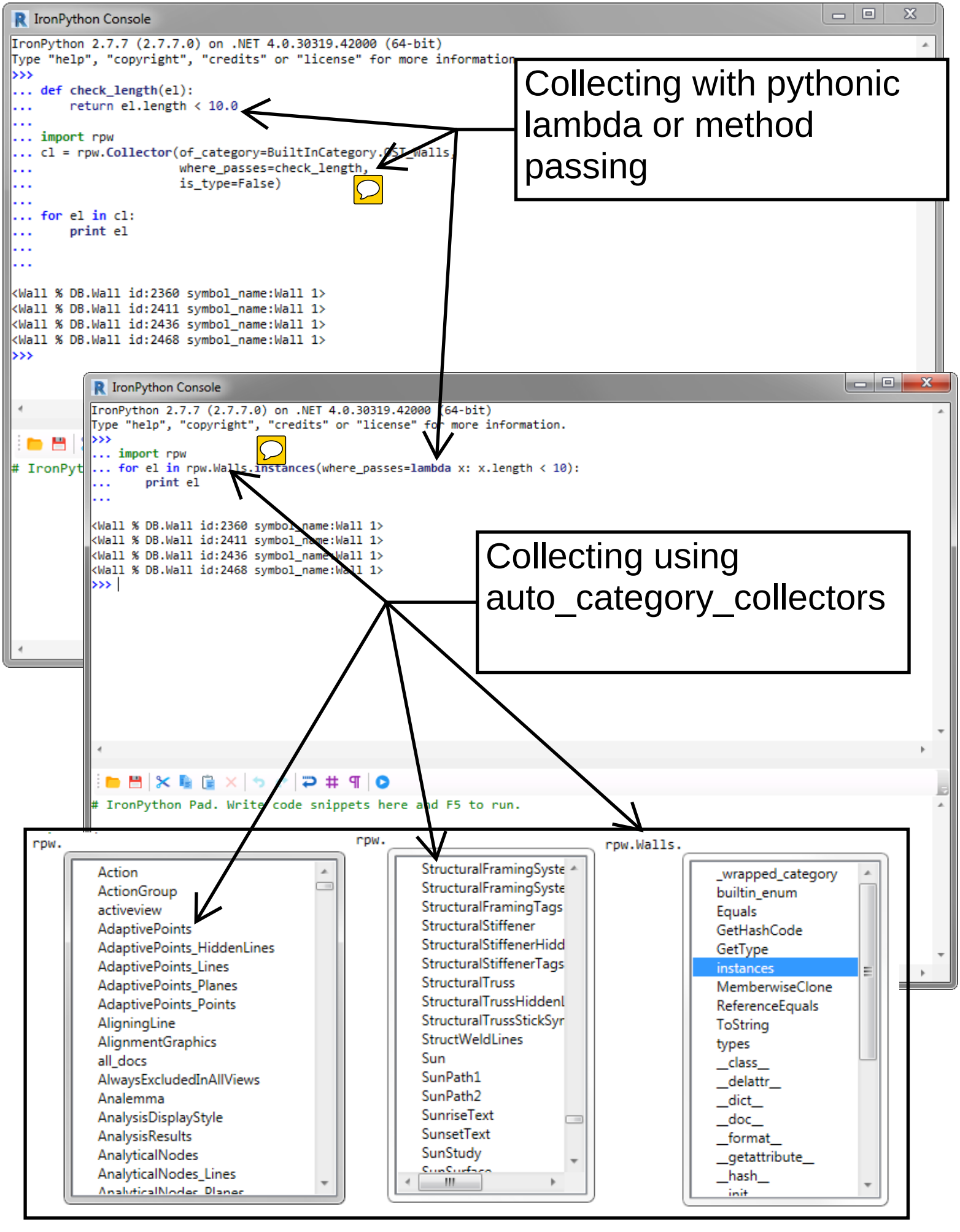
```
class Collector(BaseObjectWrapper):
    FILTER_MAP = {'of_category': 'OfCategory',
                  'of_class': 'OfClass',
                  'owned_by': 'OwnedByView',
                  'is_curve_based': 'WhereElementIsCurveDriven',
                  'is_type': {True: 'WhereElementIsElementType'
                              False: 'WhereElementIsNotElementType'},
                  'is_view_independent': {True: 'WhereElementIsViewIndependent'},
                  'where_passes': 'WherePasses',
                  'all': {True: 'WhereElementIsNotElementType'},
                  'all_types': {True: 'WhereElementIsElementType'}}

    def __init__(self, **filters):

    def filter(self, **filters):

    @property
    def elements(self):

    def __bool__(self):

    def __len__(self):

    def __repr__(self, data=None):

    def __iter__(self):

    @property
    def first(self):

    @property
    def last(self):
```

```
IronPython 2.7.7 (2.7.7.0) on .NET 4.0.30319.42000 (64-bit)
Type "help", "copyright", "credits" or "license" for more information.
>>>
... import rpw
... cl = rpw.Collector(of_category=BuiltInCategory.OST_Walls)
...
... for el in cl:
...     print el
...

<WallSymbol % DB.WallType id:1557 family_name:
<WallSymbol % DB.WallType id:1558 family_name:>
<Wall % DB.Wall id:2360 symbol_name:Wall 1>
<Wall % DB.Wall id:2385 symbol_name:Wall 1>
<Wall % DB.Wall id:2411 symbol_name:Wall 1>
<Wall % DB.Wall id:2436 symbol_name:Wall 1>
<Wall % DB.Wall id:2468 symbol_name:Wall 1>
>>>
```

Collecting all wall instances and types

```
>>>
... import rpw
... cl = rpw.Collector(all=True)
... print len(cl)
...

1751
>>>
```

Collecting all wall instances of everything


IronPython Console
IronPython 2.7.7 (2.7.7.0) on .NET 4.0.30319.42000 (64-bit)
Type "help", "copyright", "credits" or "license" for more information.
>>>
... def check_length(el):
...   return el.length < 10.0
...
... import rpw
... cl = rpw.Collector(of_category=BuiltInCategory.OST_Walls,
...   where_passes=check_length,
...   is_type=False)
...
... for el in cl:
...   print el
...
...
<Wall % DB.Wall id:2360 symbol_name:Wall 1>
<Wall % DB.Wall id:2411 symbol_name:Wall 1>
<Wall % DB.Wall id:2436 symbol_name:Wall 1>
<Wall % DB.Wall id:2468 symbol_name:Wall 1>
>>>
Collecting with pythonic lambda or method passing
IronPython Console
IronPython 2.7.7 (2.7.7.0) on .NET 4.0.30319.42000 (64-bit)
Type "help", "copyright", "credits" or "license" for more information.
>>>
... import rpw
... for el in rpw.Walls.instances(where_passes=lambda x: x.length < 10):
...   print el
...
<Wall % DB.Wall id:2360 symbol_name:Wall 1>
<Wall % DB.Wall id:2411 symbol_name:Wall 1>
<Wall % DB.Wall id:2436 symbol_name:Wall 1>
<Wall % DB.Wall id:2468 symbol_name:Wall 1>
>>>
Collecting using auto_category_collectors
IronPython Pad. Write code snippets here and F5 to run.
rpw.
Action
ActionGroup
activeview
AdaptivePoints
AdaptivePoints_HiddenLines
AdaptivePoints_Lines
AdaptivePoints_Planes
AdaptivePoints_Points
AligningLine
AlignmentGraphics
all_docs
AlwaysExcludedInAllViews
Analemma
AnalysisDisplayStyle
AnalysisResults
AnalyticalNodes
AnalyticalNodes_Lines
rpw.
StructuralFramingSyste
StructuralFramingSyste
StructuralFramingTags
StructuralStiffener
StructuralStiffenerHidd
StructuralStiffenerTags
StructuralTruss
StructuralTrussHiddenL
StructuralTrussStickSyr
StructWeldLines
Sun
SunPath1
SunPath2
SunriseText
SunsetText
SunStudy
rpw.Walls.
_wrapped_category
builtin_enum
Equals
GetHashCode
GetType
instances
MemberwiseClone
ReferenceEquals
ToString
types
__class__
__delattr__
__dict__
__doc__
__format__
__getattribute__
__hash__

Data model

So beautiful! Thank you Gui!

```
class Instance(Element):
    ...


class Symbol(Element):
    ...


class Family(Element):
    ...


class Category(Element):
    ...
```

```
>>>
... import rpw
... c = rpw.Category(BuiltInCategory.OST_Walls)
... print c
...

<WallCategory % DB.Category id:-2000011>
>>>
```

Transactions

Action or Transaction (I'm open to both, Action is shorter though :)

```
class ActionStatus:
    ...


class Action(BaseObjectWrapper):
    ...


class DryAction(Action):
    ...


class ActionGroup(BaseObjectWrapper):
    ...
```

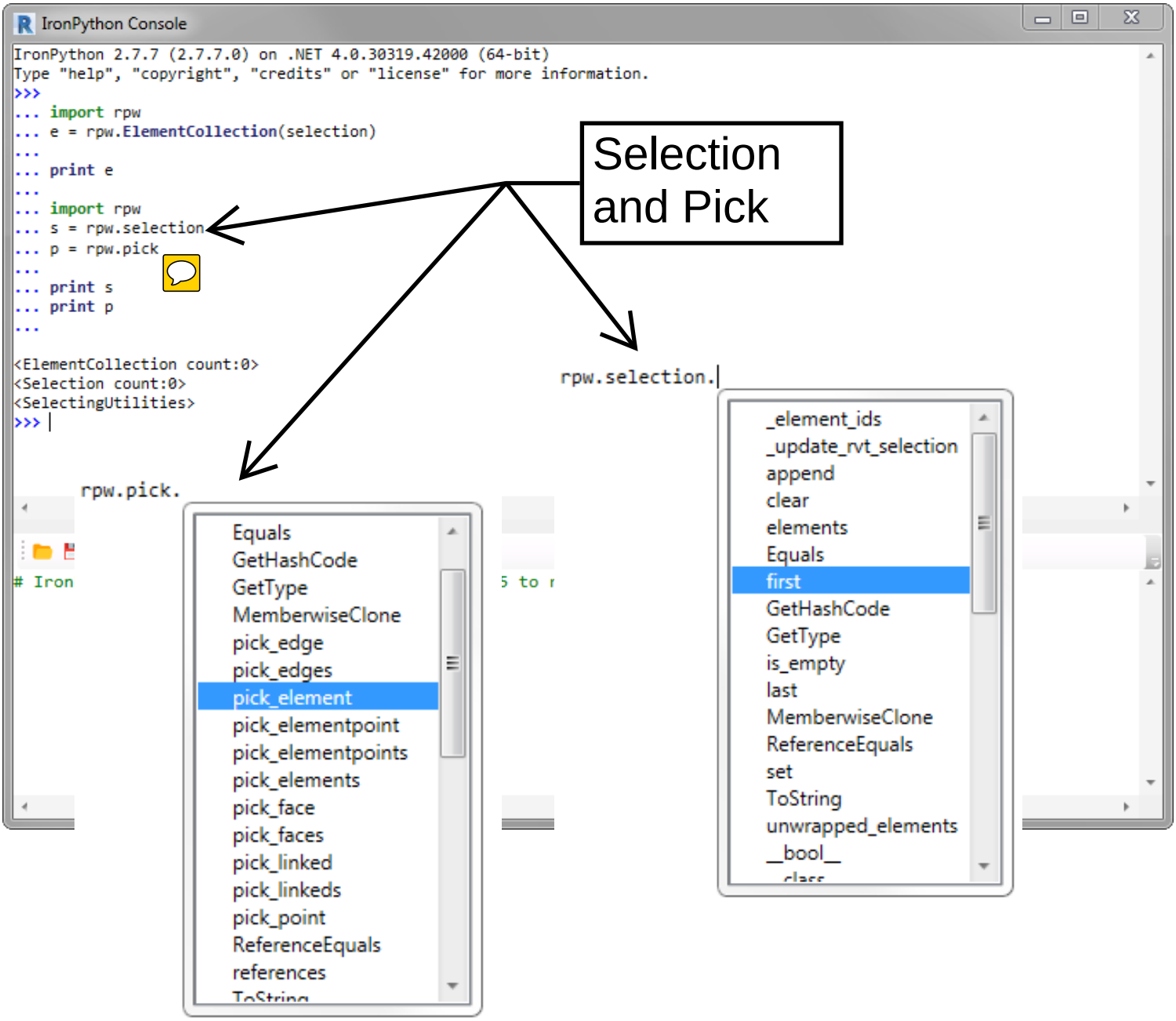
rpw.ElementCollection

```python
class ElementCollection(BaseObject):
    # def __new__(cls, mixed_list, enforce_type=None, enforce_types=None):
    def __init__(self, mixed_list, enforce_type=None, enforce_types=None):

    @property
    def unwrapped_elements(self):

    @property
    def elements(self):

    def __len__(self):

    def __iter__(self):

    def __getitem__(self, index):

    def __contains__(self, mixed_list):

    def __bool__(self):

    def __repr__(self):

    @property
    def is_empty(self):

    @property
    def first(self):

    @property
    def last(self):

    def set(self, mixed_list):

    def append(self, mixed_list):

    def clear(self):
```

Collection of elements.

Selection is a subclass of this type.

rpw.selection
rpw.pick



rpw.projectinfo

```
class ProjectInfo(BaseObjectWrapper):
    def __init__(self, api_obj):
        super(ProjectInfo, self).__init__(api_obj,
                                          enforce_type=DB.ProjectInfo)
        self.name = self._wrapped_object.Name
        self.number = self._wrapped_object.Number
        self.org_desc = self._wrapped_object.OrganizationDescription
        self.org_name = self._wrapped_object.OrganizationName
        self.status = self._wrapped_object.Status
        self.building_name = self._wrapped_object.BuildingName
        self.client_name = self._wrapped_object.ClientName
        self.issue_date = self._wrapped_object.IssueDate
        self.location = doc.PathName
```