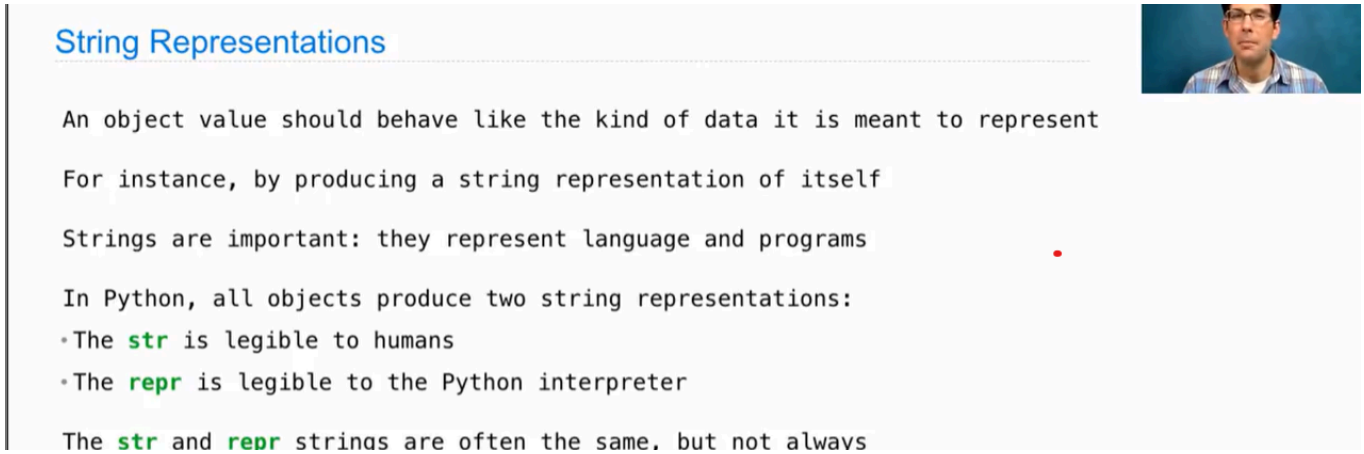
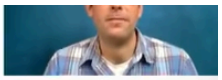
Lecture 20. Representation

String Representation



the repr string for an object

The repr String for an Object



The **repr** function returns a Python expression (a string) that evaluates to an equal object

```
repr(object) -> string

Return the canonical string representation of the object.
For most object types, eval(repr(object)) == object.
```

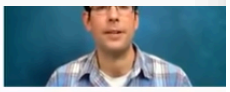
The result of calling **repr** on a value is what Python prints in an interactive session

```
>>> 12e12
120000000000000.0
>>> print(repr(12e12))
120000000000000.0
```

Some objects do not have a simple Python-readable string

```
>>> repr(min)
'<built-in function min>'
```

The str String for an Object



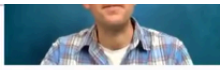
Human interpretable strings are useful as well:

```
>>> from fractions import Fraction
>>> half = Fraction(1, 2)
>>> repr(half)
'Fraction(1, 2)'
>>> str(half)
'1/2'
```

The result of calling **str** on the value of an expression is what Python prints using the **print** function:

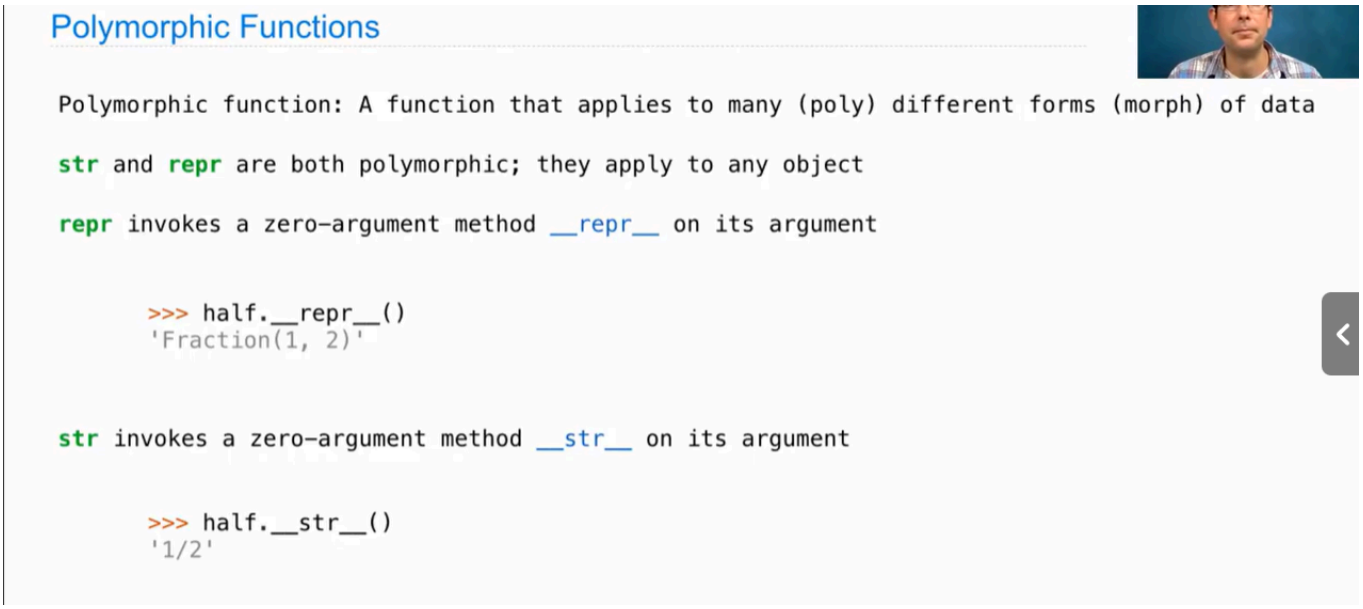
```
>>> print(half)
1/2
```

demo



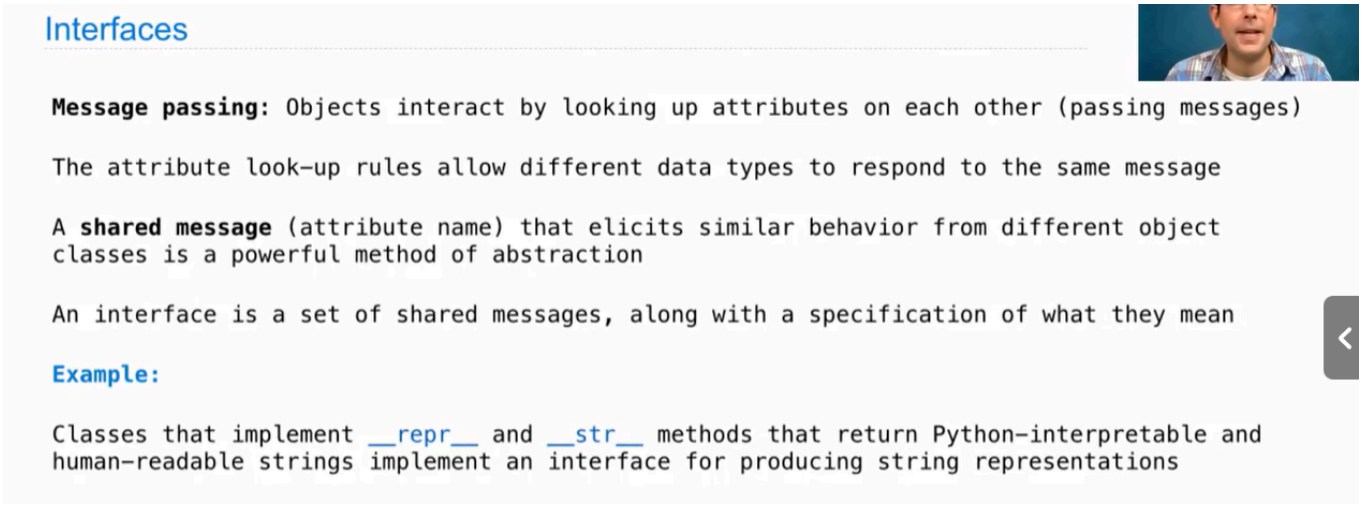
```
>>> from fractions import Fraction
>>> half = Fraction(1, 2)
>>> half
Fraction(1, 2)
>>> repr(half)
'Fraction(1, 2)'
>>> print(half)
1/2
>>> str(half)
'1/2'
>>> eval(repr(half))
Fraction(1, 2)
>>> eval(str(half))
0.5
```

Polymorphic Functions



> 简单地了来说，就是函数的多态，只不过python是轻类型语言，所以python的函数天生就是多态的。

Interfaces



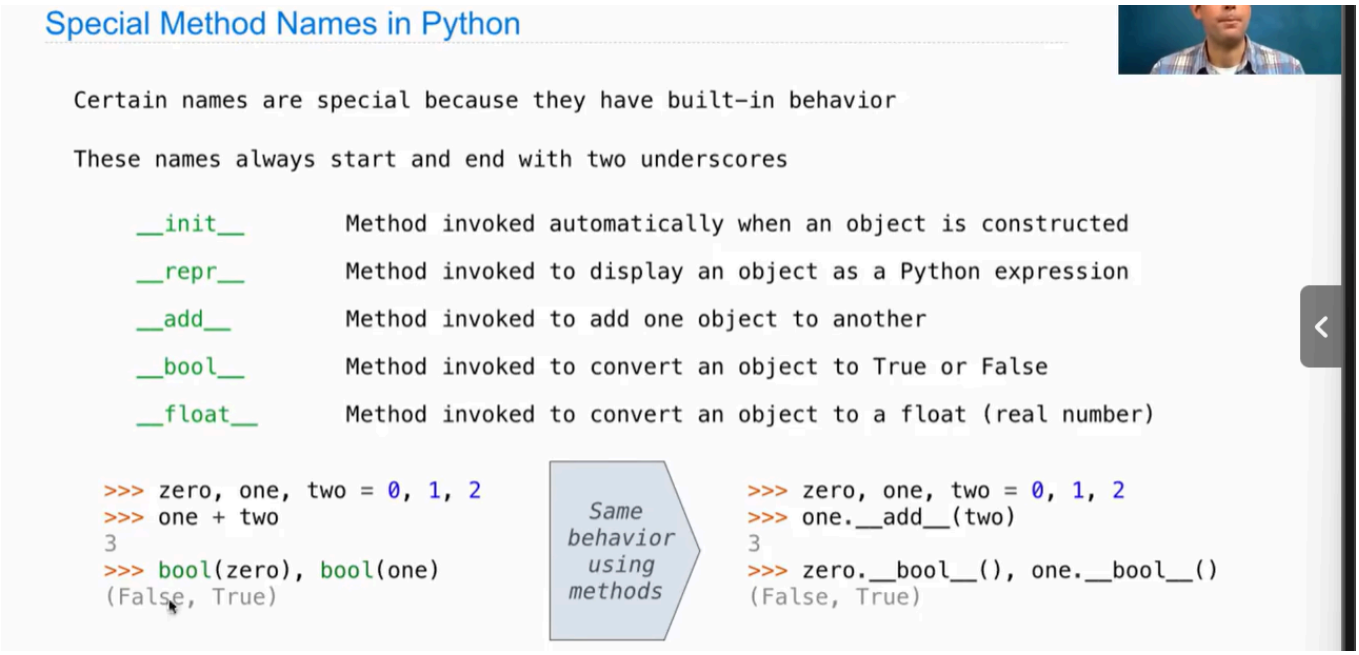
简单地去理解就是，repr和str分别是与计算机沟通的接口与与人沟通的接口，观察Ratio这个例子会理解的更加透彻

```
class Ratio:
    def __init__(self, numer, denom):
        self.numer = numer
        self.denom = denom

    def __repr__(self):
        return 'Ratio({0}, {1})'.format(self.numer, self.denom)
```

```
    def __str__(self):
        return '{0}/{1}'.format(self.numer, self.denom)
```

Special Method Names in Python



> this is another example of using interface of in order to allow user defined objects to interact to the built-in system of python