lec10-classes-and-objects

December 7, 2015

1 Classes and Objects

** Tomáš Svoboda **
Department of Cybernetics, FEE CTU in Prague
EECS, BE5B33PRG: Programming Essentials, 2015

2 nothing really new, actually

```
In [4]: a = "hello world"
        print(type(a))
```

```
<class 'str'>
```

```
In [5]: b = 5
        print(type(b))
```

```
<class 'int'>
```

3 usage

3.1 method call

```
In [6]: words_by_method =  .split()
        print(words_by_method)
```

```
['hello', 'world']
```

3.2 vs. function call

```
In [7]: words_by_function = str.split(a)
        print(words_by_function)
```

```
['hello', 'world']
```

what is difference?

```
In [12]: x = 3
         bits = x.bit_length()
         print(bits)
```

```
2
```

```
In [14]: bits = int.bit_length(8)
         print(bits)
```

```
4
```

```
In [28]: help(int)
```

```
Help on class int in module builtins:

class int(object)
 |  int(x=0) -> integer
 |  int(x, base=10) -> integer
 |
 |  Convert a number or string to an integer, or return 0 if no arguments
 |  are given.  If x is a number, return x.__int__().  For floating point
 |  numbers, this truncates towards zero.
 |
 |  If x is not a number or if base is given, then x must be a string,
 |  bytes, or bytearray instance representing an integer literal in the
 |  given base.  The literal can be preceded by '+' or '-' and be surrounded
 |  by whitespace.  The base defaults to 10.  Valid bases are 0 and 2-36.
 |  Base 0 means to interpret the base from the string as an integer literal.
 |  >>> int('0b100', base=0)
 |  4
 |
 |  Methods defined here:
 |
 |  __abs__(self, /)
 |      abs(self)
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __and__(self, value, /)
 |      Return self&value.
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __ceil__(...)
 |      Ceiling of an Integral returns itself.
 |
 |  __divmod__(self, value, /)
 |      Return divmod(self, value).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __float__(self, /)
 |      float(self)
 |
 |  __floor__(...)
 |      Flooring an Integral returns itself.
 |
 |  __floordiv__(self, value, /)
 |      Return self//value.
 |
 |  __format__(...)
```

```
|
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __getnewargs__(...)
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __hash__(self, /)
|      Return hash(self).
|
|  __index__(self, /)
|      Return self converted to an integer, if self is suitable for use as an index into a list.
|
|  __int__(self, /)
|      int(self)
|
|  __invert__(self, /)
|      ~self
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __lshift__(self, value, /)
|      Return self<<value.
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mod__(self, value, /)
|      Return self%value.
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __neg__(self, /)
|      -self
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  __or__(self, value, /)
|      Return self|value.
|
|  __pos__(self, /)
|      +self
|
```

```
|  __pow__(self, value, mod=None, /)
|      Return pow(self, value, mod).
|
|  __radd__(self, value, /)
|      Return value+self.
|
|  __rand__(self, value, /)
|      Return value&self.
|
|  __rdivmod__(self, value, /)
|      Return divmod(value, self).
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __rfloordiv__(self, value, /)
|      Return value//self.
|
|  __rlshift__(self, value, /)
|      Return value<<self.
|
|  __rmod__(self, value, /)
|      Return value%self.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __ror__(self, value, /)
|      Return value|self.
|
|  __round__(...)
|      Rounding an Integral returns itself.
|      Rounding with an ndigits argument also returns an integer.
|
|  __rpow__(self, value, mod=None, /)
|      Return pow(value, self, mod).
|
|  __rrshift__(self, value, /)
|      Return value>>self.
|
|  __rshift__(self, value, /)
|      Return self>>value.
|
|  __rsub__(self, value, /)
|      Return value-self.
|
|  __rtruediv__(self, value, /)
|      Return value/self.
|
|  __rxor__(self, value, /)
|      Return value^self.
|
|  __sizeof__(...)
|      Returns size in memory, in bytes
```

```
|
|  __str__(self, /)
|      Return str(self).
|
|  __sub__(self, value, /)
|      Return self-value.
|
|  __truediv__(self, value, /)
|      Return self/value.
|
|  __trunc__(...)
|      Truncating an Integral returns itself.
|
|  __xor__(self, value, /)
|      Return self^value.
|
|  bit_length(...)
|      int.bit_length() -> int
|
|      Number of bits necessary to represent self in binary.
|      >>> bin(37)
|      '0b100101'
|      >>> (37).bit_length()
|      6
|
|  conjugate(...)
|      Returns self, the complex conjugate of any int.
|
|  from_bytes(...) from builtins.type
|      int.from_bytes(bytes, byteorder, *, signed=False) -> int
|
|      Return the integer represented by the given array of bytes.
|
|      The bytes argument must either support the buffer protocol or be an
|      iterable object producing bytes.  Bytes and bytearray are examples of
|      built-in objects that support the buffer protocol.
|
|      The byteorder argument determines the byte order used to represent the
|      integer.  If byteorder is 'big', the most significant byte is at the
|      beginning of the byte array.  If byteorder is 'little', the most
|      significant byte is at the end of the byte array.  To request the native
|      byte order of the host system, use `sys.byteorder' as the byte order value.
|
|      The signed keyword-only argument indicates whether two's complement is
|      used to represent the integer.
|
|  to_bytes(...)
|      int.to_bytes(length, byteorder, *, signed=False) -> bytes
|
|      Return an array of bytes representing an integer.
|
|      The integer is represented using length bytes.  An OverflowError is
|      raised if the integer is not representable with the given number of
|      bytes.
```

```
 |
 |      The byteorder argument determines the byte order used to represent the
 |      integer.  If byteorder is 'big', the most significant byte is at the
 |      beginning of the byte array.  If byteorder is 'little', the most
 |      significant byte is at the end of the byte array.  To request the native
 |      byte order of the host system, use `sys.byteorder' as the byte order value.
 |
 |      The signed keyword-only argument determines whether two's complement is
 |      used to represent the integer.  If signed is False and a negative integer
 |      is given, an OverflowError is raised.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  denominator
 |      the denominator of a rational number in lowest terms
 |
 |  imag
 |      the imaginary part of a complex number
 |
 |  numerator
 |      the numerator of a rational number in lowest terms
 |
 |  real
 |      the real part of a complex number
```

4 method vs function

method is associated with a specific object

```
a.split()
```

```
str.split(a)
```

`str` is a module, `a` is here an input parameter of the `split` function

5 method vs function for lists

```
In [15]: print(words_by_function)
         list.append(words_by_function,'!')
         print(words_by_function)
```

```
['hello', 'world']
['hello', 'world', '!']
```

```
In [16]: print(words_by_method)
         words_by_method.append('!')
         print(words_by_method)
```

```
['hello', 'world']
['hello', 'world', '!']
```

6 user defined type - charging particles

** application programming interface **

operation	description
`Charge(x0,y0,q0)`	a new charged centered at (x0, y0) with charge q0
`c.potential_at()`	electric potential of charge `c` at point (x,y)
`str(c)`	a string represenation of charge c

[Constructor visualization](http://www.pythontutor.com/visualize.html#code=class+Charge%3A%0A++++def+**init**(s
frontend.js&cumulative=false&heapPrimitives=false&textReferences=false&py=3&rawInputLstJSON=%5B%5D&curInstr=
[potential_at](http://www.pythontutor.com/visualize.html#code=class+Charge%3A%0A++++def+**init**(self,+x0,+y0,+
+self.x%0A+++++++++dy+%3D+y+-+self.y%0A+++++++++r+%3D+(dxdx+%2B+dydy%29**(1/2%29%0A+++++
frontend.js&cumulative=false&heapPrimitives=false&textReferences=false&py=3&rawInputLstJSON=%5B%5D&curInstr=
[__str__](http://www.pythontutor.com/visualize.html#code=class+Charge%3A%0A++++def+**init**(self,+x0,+y0,+q0%
+self.x%0A+++++++++dy+%3D+y+-+self.y%0A+++++++++r+%3D+(dxdx+%2B+dydy%29**(1/2%29%0A+++++
frontend.js&cumulative=false&heapPrimitives=false&textReferences=false&py=3&rawInputLstJSON=%5B%5D&curInstr=

7 notebook config

ignore the cells below

In [1]:
```
from notebook.services.config import ConfigManager
cm = ConfigManager()
cm.update('livereveal', {
              'theme': 'White',
              'transition': 'None',
              'start_slideshow_at': 'selected',
              'width': 1024,
              'height': 768,
              'minScale': 1.0
})
```

Out[1]:
```
{'height': 768,
 'minScale': 1.0,
 'start_slideshow_at': 'selected',
 'theme': 'White',
 'transition': 'None',
 'width': 1024}
```

In [2]:
```
%%HTML
<style>
.reveal #notebook-container { width: 90% !important; }
.CodeMirror { max-width: 100% !important; }
pre, code, .CodeMirror-code, .reveal pre, .reveal code {
    font-family: "Consolas", "Source Code Pro", "Courier New", Courier, monospace;
}
pre, code, .CodeMirror-code {
    font-size: inherit !important;
}
.reveal .code_cell {
    font-size: 130% !important;
    line-height: 130% !important;
}
</style>
```

```
<IPython.core.display.HTML object>
```