Classes and Objects

Tomáš Svoboda

Department of Cybernetics, FEE CTU in Prague

EECS, BE5B33PRG: Programming Essentials, 2016

from namedtuple to class

In [16]:

```
from collections import namedtuple
```

In [17]:

```
Point = namedtuple('Point','x y')
p1 = Point(x=3,y=4)
print(p1, p1.x, p1.y)
print(type(p1))
```

```
Point(x=3, y=4) 3 4
<class '__main__.Point'>
```

In [18]:

```
class Point2D:
    def __init__(self,x_coord,y_coord):
        self.x = x_coord
        self.y = y_coord
```

In [19]:

```
p1 = Point2D(3,4)
print(p1, p1.x, p1.y)
```

```
<__main__.Point2D object at 0x10bf6a550> 3 4
```

Can we change p1? Can we shift it? Can we p = p1 + p2 somehow?

In [20]:

```
def add_points(p1,p2):
    p = []
    for i in range(len(p1)):
        p.append(p1[i]+p2[i])
    return(p)

def point_shift(p,dx,dy):
    p[0] += dx
    p[1] = p[1] + dy

p1 = [3,4]
```

```
# add_points(p1,p1)
point_shift(p1,10,10)
print('p1 is now', p1)
```

```
p1 is now [13, 14]
```

a new, user defined datatype, class

In [71]:

```
class PointNd:
    def __init__(self, *arg):
        if len(arg) == 1:
            self.c = list(arg[0])
        else:
            self.c = []
            for i in range(len(arg)):
                self.c.append(arg[i])
    def shift(self, coord_shift):
        for i in range(len(self.c)):
            self.c[i] += coord_shift[i]
    def __add__(self,other): # overloading + operator
        res = []
        for i in range(len(self.c)):
            res.append(self.c[i]+other.c[i])
        return PointNd(res)
    def __sub__(self, other):
        res = []
        for i in range(len(self.c)):
            res.append(self.c[i]-other.c[i])
        return PointNd(res)
    def __str__(self):
        return(str(self.c))
    def __abs__(self):
        res = 0
        for ele in self.c:
            res += ele**2
        return res**(1/2)
```

In [75]:

```
ps1 = PointNd(['a','b'])
print(ps1-ps1)
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call last)
<ipython-input-75-139e333bba3f> in <module>()
      1 ps1 = PointNd(['a','b'])
----> 2 print(ps1-ps1)

<ipython-input-71-506ab0f40a1e> in __sub__(self, other)
     18         res = []
     19         for i in range(len(self.c)):
---> 20             res.append(self.c[i]-other.c[i])
     21         return PointNd(res)
     22     def __str__(self):

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

In [67]:

```
p1 = PointNd(3,4,4)
p2 = PointNd([2,3,10])
p3 = PointNd(1,0,0)
print(abs(p1+p2))
abs(p3)
print(p1-p2)
```

```
16.431676725154983
```

Out[67]:

```
1.0
```

```
[1, 1, -6]
```

`print(p3)`, printing nicely

More about special methods: http://www.diveintopython3.net/special-method-names.html, http://www.python-course.eu/python3_magic_methods.php, https://docs.python.org/3.4/reference/datamodel.html

nothing really new, actually

In [54]:

```
a = "hello world"
print(type(a))
```

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

In [10]:

```
b = 5
print(type(b))
```

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

In [56]:

```
l = list([2,3,4])
print(type(l))
```

```
<class 'list'>
```

actually, (almost) everything is class in Python

class instance - object

In [58]:

In [58]:

```
a
type(a)
```

Out[58]:

'hello world'

Out[58]:

str

In [12]:

```
c = "hi"
type(c)
```

Out[12]:

str

In [ ]:

```

```

---

usage

method call

In [60]:

```
words_by_method = a.split()
print(words_by_method)
```

['hello', 'world']

vs. function call

In [61]:

```
words_by_function = str.split(a)
print(words_by_function)
```

['hello', 'world']

what is difference?

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

method vs function for lists

In [63]:

```
print(words_by_function)
list.append(words_by_function,'!')
print(words_by_function)
```

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

In [64]:

```
print(words_by_method)
words_by_method.append('!')
print(words_by_method)
```

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

(Point) charge

application programming interface

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

Potential created by charge Q at a distance r from charge: $V = \frac{1}{4\pi\epsilon_0}\frac{Q}{r}$

Constructor visualization, potential_at, __str__

In [70]:

```
class Charge:
    def __init__(self, pos, q0):
        self.pos, self.q = pos, q0

    def potential_at(self, pos):
        COULOMB = 8.99e09
        r = abs(self.pos-pos)
        if r < 1e-19: # Avoid division by 0
            return float('inf')
```

```
        return COULOMB * self.q / r

    def __str__(self):
        result = str(self.q) + ' at '
        result += str(self.pos)
        return result
q1 = Charge(PointNd([3,4]),10)
print(q1)
pos2 = PointNd([5,5])
pos3 = PointNd([3,3])
pos4 = pos2+pos3
p1 = q1.potential_at(pos2)
print('potential of',q1, 'in',  pos2, 'is', p1)
```

```
10 at [3, 4]
potential of 10 at [3, 4] in [5, 5] is 40204502235.44622
```

In [ ]:

```
```

What about using `Point2D` class?

notebook config

ignore the cells below

In [1]:

```
from notebook.services.config import ConfigManager
cm = ConfigManager()
cm.update('livereveal', {
              'theme': 'White',
              'transition': 'none',
            'scroll': True,
            'center': True,
              'start_slideshow_at': 'selected',
              'width': '100%',
              'height': '100%',
              'minScale': 1.0
})
```

Out[1]:

```
{'center': True,
 'controls': True,
 'height': '100%',
 'minScale': 1.0,
 'mouseWheel': True,
 'progress': True,
 'scroll': True,
 'slideNumber': True,
 'start_slideshow_at': 'selected',
 'theme': 'White',
 'transition': 'none',
 'width': '100%'}
```

In [6]:

```
%%HTML
<style>
.reveal # notebook-container { width: 100% !important; }
.reveal pre { width: 100% !important; }
# .prompt { display: None !important; }
.CodeMirror { max-width: 100% !important; }
# .CodeMirror .cm-s-ipython { width: 100% !important; }
# .CodeMirror-lines { width: 100% !important; }
.container {width:95% !important; }
pre, code, .CodeMirror-code, .reveal pre, .reveal code {
    font-family: "Consolas", "Source Code Pro", "Courier New", Courier, monos
pace;
}
pre, code, .CodeMirror-code {
    font-size: inherit !important;
}
.reveal .code_cell {
    font-size: 130% !important;
    line-height: 130% !important;
}
</style>
```