Classes and Objects

Tomáš Svoboda

Department of Cybernetics, FEE CTU in Prague

EECS, BE5B33PRG: Programming Essentials, 2016

from namedtuple to class

In [13]:

```
from collections import namedtuple
```

In [18]:

```
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 [37]:

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

In [38]:

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

```
<__main__.Point2D object at 0x107d032b0> 3 4
```

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

In [31]:

```
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]
```

---

nothing really new, actually

In [3]:

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

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

In [4]:

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

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

actually, (almost) everything is class in Python

class instance - object

In [5]:

```
type(a)
```

Out[5]:

```
str
```

In [6]:

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

Out[6]:

```
str
```

In [ ]:

```

```

---

usage

usage

method call

In [7]:

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

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

vs. function call

In [8]:

```
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 [9]:

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

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

In [10]:

```
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__`

notebook config

ignore the cells below

In [11]:

```
from notebook.services.config import ConfigManager
cm = ConfigManager()
cm.update('livereveal', {
              'theme': 'White',
              'transition': 'linear',
            'scroll': True,
            'center': True,
              'start_slideshow_at': 'selected',
              'width': 1024,
              'height': 768,
              'minScale': 1.0
})
```

Out[11]:

```
{'center': True,
 'controls': True,
 'height': 768,
 'minScale': 1.0,
 'mouseWheel': True,
 'progress': True,
 'scroll': True,
 'slideNumber': True,
 'start_slideshow_at': 'selected',
 'theme': 'White',
 'transition': 'linear',
 'width': 1024}
```

In [12]:

```
%%HTML
<style>
.reveal # notebook-container { width: 100% !important; }
 CodeMirror { max-width: 100% !important; }
```

```
.CodeMirror { max-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>
```