

关于Swampy.Turtle中画图函数

在TurtleWorld模块之中，可以通过两类写法控制乌龟：

1. `bob.fd(100)` ----> `Turtlename.functionname()` 对象.方法(形参)
2. `fd(bob, 100)` ----> `functionname(Turtlename)` 方法(形参)

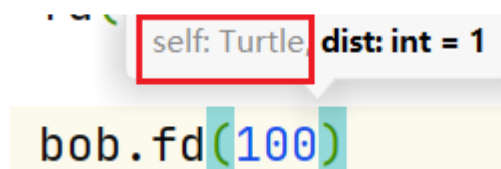
1. 对象.方法(形参)

第1种方式易于理解，原因是Turtle类之中实现了fd方法，则由Turtle类创建出的对象可以使用fd方法。

和普通函数相比，在类中定义的成员方法有一点不同(类方法、静态方法此处不讨论)，就是**第一参数永远是类的本身实例变量self**，并且调用时，不用传递该参数。例如下图([TurtleWorld.py, line 176](#))中的类内成员函数

```
176 def fd(self, dist=1):
177     """Moves the turtle foward by the given distance."""
178     x, y = self.x, self.y
179     p1 = [x, y]
180     p2 = self.polar(x, y, dist, self.heading)
181     self.x, self.y = p2
182
183     # if the pen is down, draw a line
184     if self.pen and self.world.exists:
185         self.world.canvas.line([p1, p2], fill=self.pen_color)
186     self.redraw()
```

下图使用工具验证：(Pycharm快捷键 `ctrl + p`)

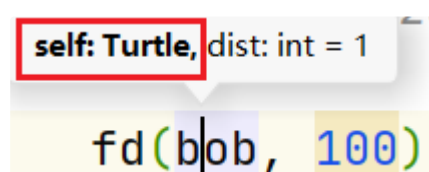


2. 方法(形参)

第2种方式，通过阅读TurtleWorld模块的源码，可以找到此模块的作者在下图处([TurtleWorld.py, line 225](#))定义了全局变量，这些全局变量(fd/bk/lt...)为Turtle类中成员方法的引用。由于均为全局变量，因此可以在 `__main__` 函数之中直接使用，并且这些全局变量(fd/bk/lt...)的值也没有被改变，也无需再使用 `global` 声明

```
225 """Add the turtle methods to the module namespace
226 so they can be invoked as simple functions (not methods).
227 """
228 fd = Turtle.fd
229 bk = Turtle.bk
230 lt = Turtle.lt
231 rt = Turtle.rt
232 pu = Turtle.pu
233 pd = Turtle.pd
234 die = Turtle.die
235 set_color = Turtle.set_color
236 set_pen_color = Turtle.set_pen_color
237
```

下图粗体说明光标位置处对应第一个形参self,形参类型为Turtle



3. 总结

下面举一个同类型的例子，进一步说明：

```

class MethodClass:
    def test(self):
        pass

# 设置全局变量，引用MethodClass类中的test方法
global_method = MethodClass.test
# 创建MethodClass类实例对象class_instance
class_instance = MethodClass()
# 方式1
class_instance.test()
# 方式2
global_method(class_instance)

```

同时借助一些工具，观察变量的类型，可以更好地理解python中万物皆对象的思想

```

> class_instance = MethodClass <__main__.MethodClass object at 0x0000018A4229E850>
Special Variables
  01 __name__ = {str} '__main__'
> __builtins__ = {module} <module 'builtins' (built-in)>
  01 __doc__ = {str} 'Automatically created module for IPython interactive environment'
> __builtin__ = {module} <module 'builtins' (built-in)>
  01 __loader__ = {NoneType} None
  01 __spec__ = {NoneType} None
  01 __package__ = {NoneType} None
> sys = {module} <module 'sys' (built-in)>
> MethodClass = {type} <class '__main__.MethodClass'>
> global_method = function <function MethodClass.test at 0x0000018A4228DAF0>
  01 __ = {str} ''

```