>>>

PyShare Day4

@iihciyekub

Step by step

LYJ8512@126.com

工欲善其事，必先利其器

Life is short, I use Python

人生苦短,我用python

古人說的~

还真得一步一脚印,慢慢来

器欲尽其能，必先得其法

厉害是 python,不是使用的 python 的人,

学无止境,

只有理解代码术语, python 的基本语法,语句, 才能用好更多的利(类)器(库)

古人接着又說了~

D4 分享内容速览

1

0.有趣的,蒙特卡洛方法

通过代码复习基本语句,语法,

- *估算π!*

>>> 测试一个,对编程,掌握有多少

1.编程术语扫盲

术语 → 理解术语 → 思想

- *理解他人代码,*
- *描述代码,分享代码,编写文档*
- *与他人协作,精准描述问题*

2

3

2.面向对象思想

把大象塞进冰箱,

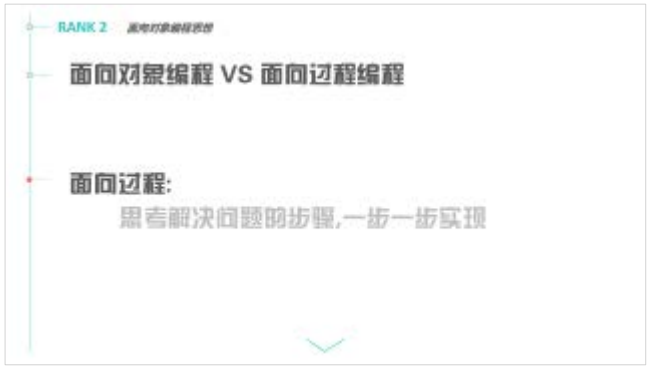
- *理解各种术语*
- *感受面向对象思想的妙处*

>>> 做个小练习,感受一下吧

4

RANK 0

RANK 1



RANK 2

蒙特卡罗（Monte Carlo）法

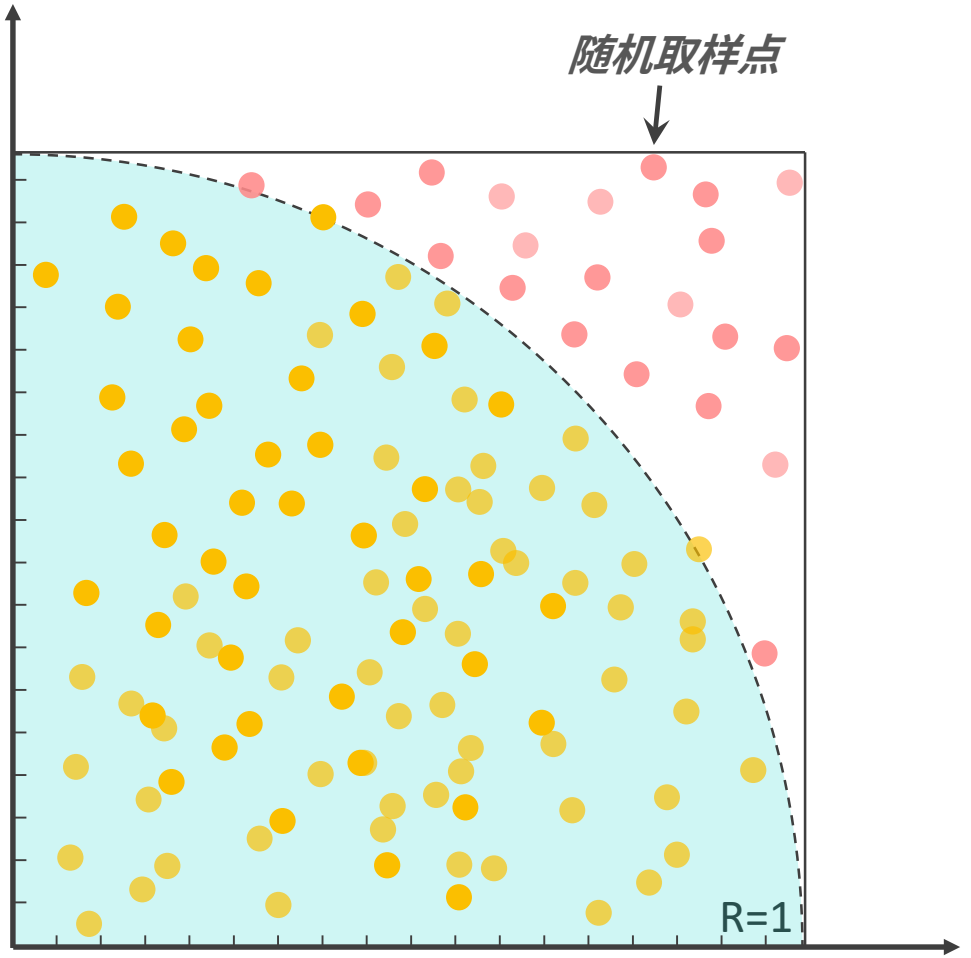
诞生于上个世纪40年代美国的"曼哈顿计划"，名字来源于赌城蒙特卡罗，象征概率。

蒙特卡罗方法是一种计算方法。

原理是通过大量随机样本，去了解一个系统，进而得到所要计算的值。

蒙特卡罗（Monte Carlo）法

$$\frac{1}{4}\pi r^2 = \frac{\text{●}\ \text{数量}}{[\text{●} + \text{●}]\ \text{数量}}$$



π =?

if 语句

```
a = 0
if a > 0:
    print('do something')
```

条件语句

注意缩进

For 语句

```
for i in [1,2,3,4,5,6]:
    print("do something")
```

临时变量,随意取名

可迭代对象

注意缩进

import 关键词 → 导入类库

```
1 import random
2
3 # 可能简单的理解成生成指定的随机样本
4 random.seed(1)
5
6 # [0,1)之间的随机样本,加了上面的方法后,每次数值是一样的
7 for i in range(10):
8     print(random.random())
```

可迭代对象,可理解成

[0,1,2,3,...,9]

生成[0,1)的随机数

>>> 挑战一下:实现代码

RANK 0 *Pi值估算,代码实现:*

N1：产生一个随机样本点,(数量越多,估算就会越准确)
比如100000

N2：利用for+if 语句 重复判断(次数=随机样本点数量)
比如100000

N3：计算落在圆内的样本点占总数量的4倍比率 p

提示:

@@,你可能经常听到以下这些名词:

对象

继承

函数

lambda

装饰器

递归

字段

封装

迭代器

静态属性

闭包

类

多态

形参实参

实例化

方法

@@,听到这,你可能已经不想学了:

或者已经被吓到了,错认为门槛很高

但是,这仅仅是表象而已,
表象往往是很容易欺骗人,
事实上,python的学习成本非常低,

Python 代码的可阅读性较高...

只要我们理解了这些术语

下面,我们通过代码去理解这些乱78糟的术语

请记住

代码→现实世界的映射

这些术语,你总能在现实世界中找到与之对应的关系

1.类的作用,就是为了分类

服装

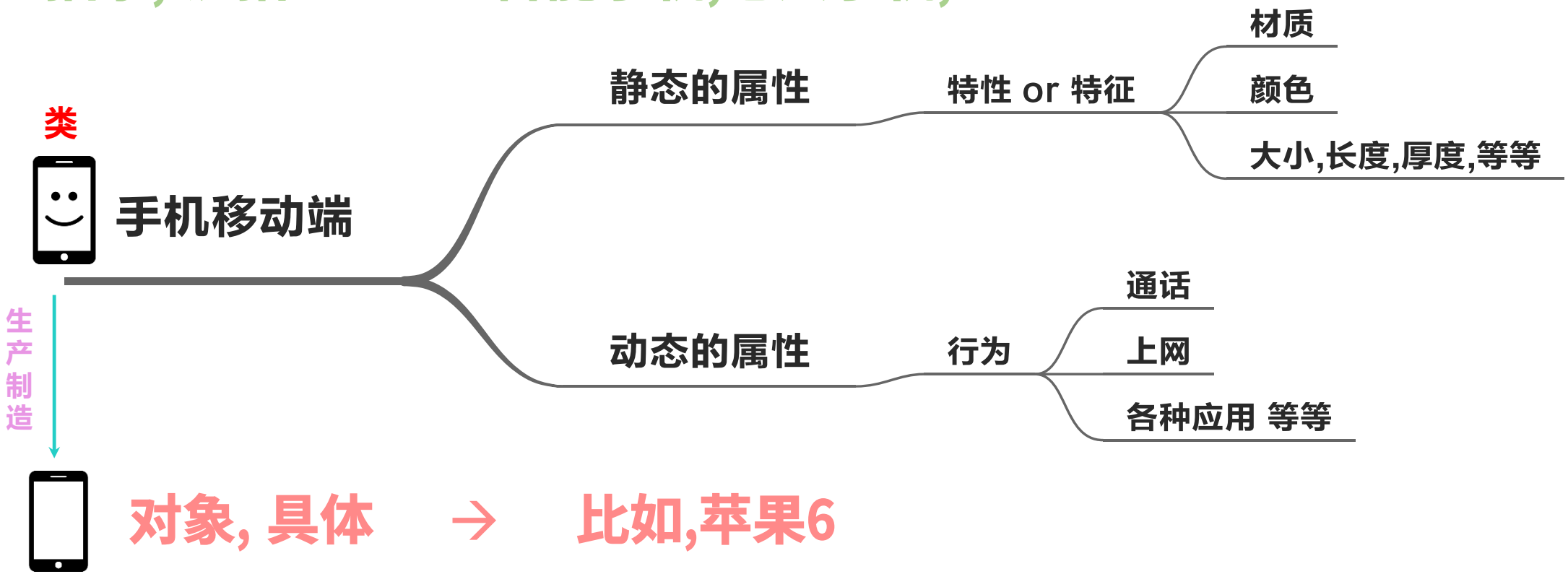
手机

相机

RANK 1 *现实生活角度,理解: 类 & 对象*

1.类,用来描述具有相同特征和行为的抽象

抽象, 泛指 → 智能手机,老人手机,



对象, 具体 → 比如,苹果6

RANK 1 *现实生活角度,理解: 类 & 对象*

```
1 class Mobile(object):
2       sytem = "ios"
3       color = 'w'
4
5       def callSomeOne(self):
6           print("call...")
7
8
9 appleM = Mobile()
```

生产制造

类名

字段:

描述类的特性 or 特征

方法:

描述类的行为

实例化:

得到一个对象 appleM

小练习:

随意创建类,熟悉一下语法结构

```
1 class Mobile(object):
2     sytem = "ios"
3     color = 'w'
4
5     def callSomeOne(self):
6         print("call...")
7
8
9 appleM = Mobile()
```

面向对象编程 VS 面向过程编程

面向过程:

思考解决问题的步骤,一步一步实现

面向对象编程

构成问题的事物分解成对象

描述某个事物在解决问题过程中的行为

面向对象编程

面向对象三大特性:封装,继承,多态

重用性,灵活性,扩展性

如果把大象塞进冰箱里,总共需要多少步呢?

面向过程思想,实现….

```
1 print("打开冰箱")
2 print("把大象塞进冰箱")
3 print("关闭冰箱")
```

代码量大,无重用性,需求改变时,修改量巨大..

如果把大象塞进冰箱里,总共需要多少步呢?

面向对象思想,实现….

1.创建类,添加字段属性,方法,让它具有特定的行为

2.实例化出一个对象(创建对象)

3.通过组合对象的行为,解决问题

编写代码,感受这种编程思想

小练习：

创建冰箱类，
实现把大象放进冰箱

```python
# 1 创建类,添加字段,属性,方法,让它具有特定的行为
class Refrigerator():

    def open(self):
        print("打开冰箱门")

    def close(self):
        print("关闭冰箱门")

    def putIn(self, thing):
        print(f"把{thing}塞进冰箱")

# 2 实例化出一个对象
r=Refrigerator()

# 3 通过组合对象的行为,解决问题
r.open()
r.putIn("大象")
r.close()
```

封装

- 保护代码
- 屏蔽复杂性
- 实现方法重用

封装

封装是面向对象编程的第一步….

试想一下…

满屋子一堆玩具,散落在地面上,很乱

这时候,我们可以对玩具进行分类,不同类放在不同箱子里

继承

父类,子类,

子类继承于父类后,拥有父类的全部特性

小练习：

创建继承于冰箱的洗衣机类，

实现把鲸鱼放进洗衣机

```python
# 1 创建类,添加字段,属性,方法,让它具有特定的行为
class Refrigerator():

    def __init__(self):
        self.name='冰箱'

    def open(self):
        print(f"打开{self.name}门")

    def close(self):
        print(f"关闭{self.name}门")

    def putIn(self, thing):
        print(f"把{thing}塞进冰箱")

    # 3 将行为组合,并封装成方法,解决问题
    def todo(self,thing):
        self.open()
        self.putIn(thing)
        self.close()


class Washer(Refrigerator):
    def __init__(self):
        self.name='洗衣机'



# 2 实例化出一个对象
r=Refrigerator()
r.todo("大象")

# 4 实例化出一个子对象
p=Washer()
p.todo('鲸鱼')
```

Thank You

PyShare D4