清華大學
Tsinghua University

小伙伴计划暑期学习营——零基础Python入门

第三讲：Python基础语法
——面向对象、模块

张智帅 电子系

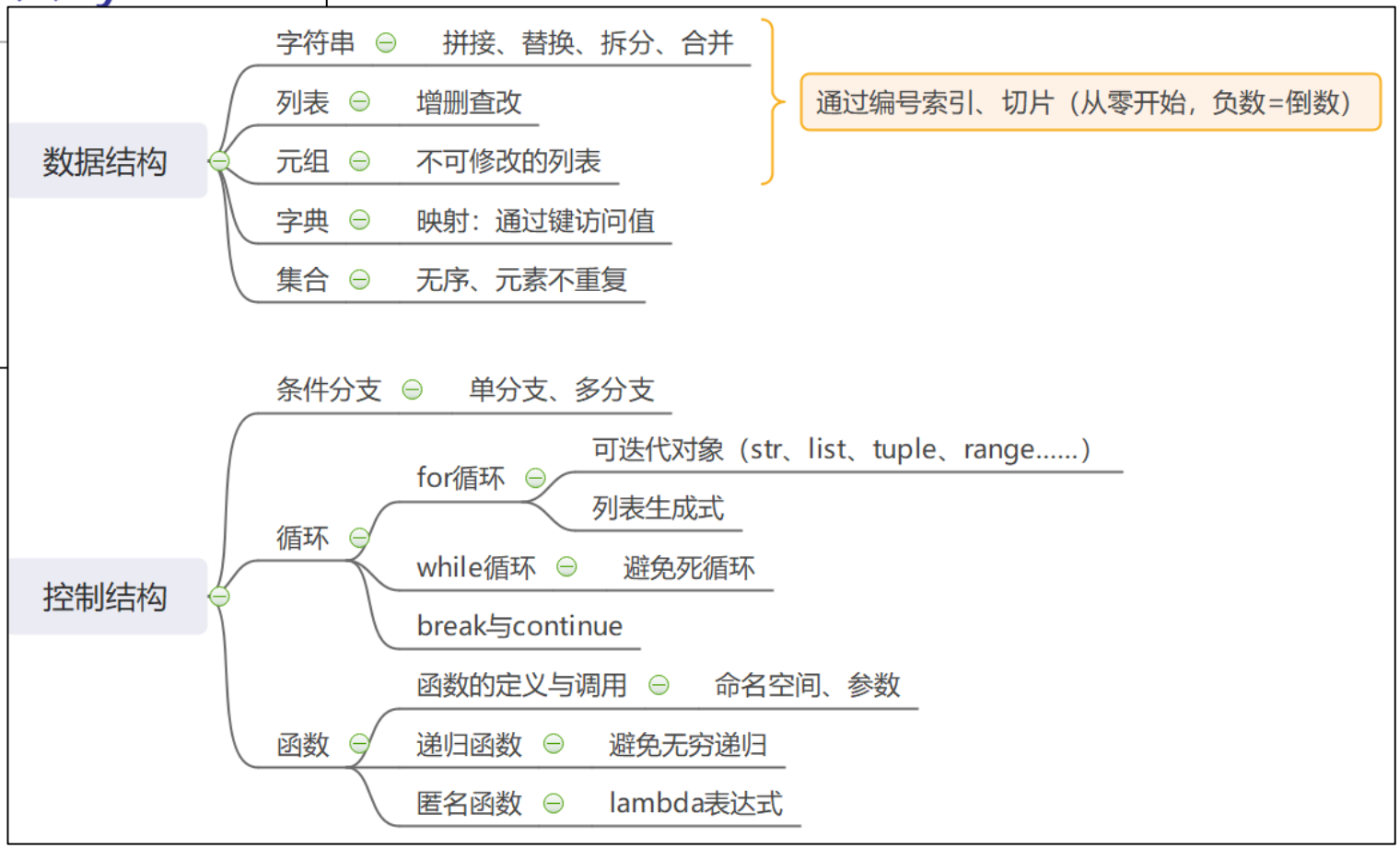
清华大学学生学业与发展指导中心

2019-2020学年夏季学期

第一、二讲总结

第一讲：认识Python

- 宏观认识
- 直观认识
- 微观认识



第三讲：面向对象、模块

- 面向对象
  - 面向对象编程思想
  - 创建自定义类型
- 模块

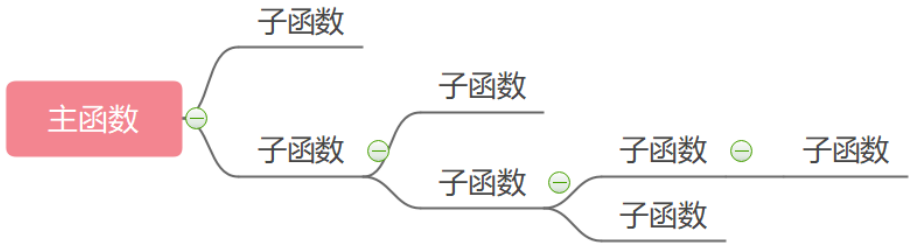
Python3 智能模式

```python
class Solution:
    def numJewelsInStones(self, J: str, S: str) -> int:
        pass
```

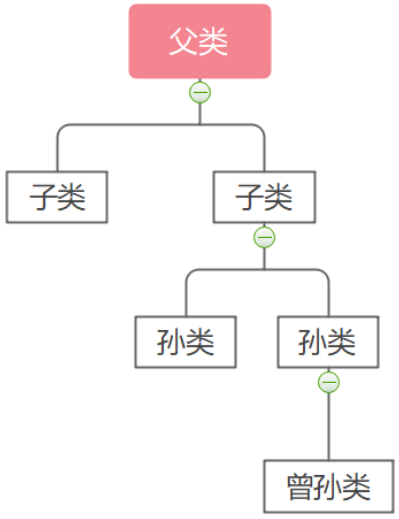
面向对象编程思想

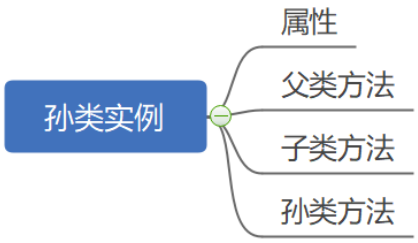
- 编程范式：函数式编程、面向过程编程、面向对象编程.....
- 函数式编程（FP）：描述映射

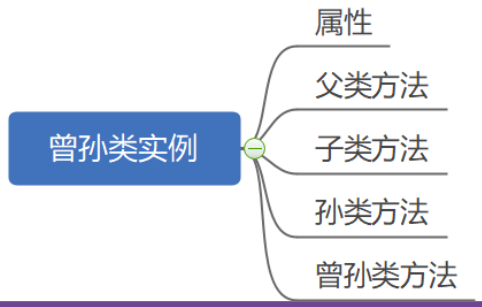
- 面向过程（POP）：按照步骤，逐步实现



- **面向对象**（OOP）：定义对象，然后描述对象的行为。
  - **封装、继承、多态**
  - **易维护、易复用、易扩展**







万物皆对象

- 纯面向对象：于 Python 而言，万物皆对象。

```
def show():
    pass


s = "tsinghua"
b = 124
print(type(s))  # 字符串是 str 类的对象
print(type(b))  # 数字是 int 类的对象
print(type(show))  # 自定义函数是 function 类的对象
print(type(print))  # print函数是 builtin_function_or_method 类的对象
```

```
<class 'str'>
<class 'int'>
<class 'function'>
<class 'builtin_function_or_method'>
```

- 对象的三大基本特征：**值**（value）、**类型**（type）、 **id**（identity）

```
a = [1, 2, 3, 4, 5, 6, 7, 8]
print(a)  # value
print(type(a))  # type
print(id(a))  # identity
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
<class 'list'>
1859573442376
```

面向对象编程基本要素

- 类和对象是Python的核心概念
  - 类：是抽象的模板，是具有相同属性和方法的一类事物
  - 对象 / 实例：这一类事物中具体的一个

类属性

类

类方法

对象 / 实例

创建自定义类型

- 自定义**类**（class）
  - 类属性、类方法；
  - 公有变量、私有变量

```
class Hero:  # 定义 Hero 类

    name = ""  # name 属性，默认为公有变量
    __level = 45  # 两个下划线作为前缀，表示私有变量

    def set_info(self, name, level):
        self.name = name  # 给对象的 name 属性赋值
        self.__level = level  # 给对象的 __level 属性赋值

    def GANK(self):  # 定义 Hero 类的GANK方法
        print("{} 级的 {} 前来支援！".format(self.__level, self.name))

    def use_skill(self, number):  # 定义 Hero 类的 use_skill 方法
        print("{} 级的 {} 使用了 {} 技能".format(self.__level, self.name, number))
```

类属性

类方法

- self：是所有类方法位于首位、默认的特殊参数，代表“这个对象”

实例化

- **实例**（Instance）是根据类创建出来的一个个具体的“对象”
  - 调用类方法、访问类属性
  - 每个对象都拥有相同的方法，但各自的属性可能不同

```
a = Hero()  # 对象 a 是 Hero 类的一个实例
b = Hero()  # 对象 b 也是 Hero 类的一个实例
c = Hero()  # 对象 c 也是 Hero 类的一个实例

# 调用类的方法
a.set_info("Batman", 12)
b.set_info("Superman", 13)
c.set_info("Spiderman", 15)
a.use_skill(1)
b.use_skill(2)
c.use_skill(3)
```

- 能否在实例化的时候就设置属性?

构造方法

- **__init__方法**：在创建实例的时候，就给属性赋值

```
class Hero:  # 定义 Hero 类
    def __init__(self, name, level):  # 构造方法
        self.name = name  # 给对象的 name 属性赋值
        self.__level = level  # 给对象的 __level 属性赋值

    def GANK(self):  # 定义 Hero 类的GANK方法…
```

```
a = Hero("Batman", 12)
b = Hero("Superman", 13)
c = Hero("Spiderman", 15)
```

- 对比：

```
class Hero:  # 定义 Hero 类

    name = ""  # name 属性，默认为公有变量
    __level = 45  # 两个下划线作为前缀，表示私有变量

    def set_info(self, name, level):
        self.name = name  # 给对象的 name 属性赋值
        self.__level = level  # 给对象的 __level 属性赋值

    def GANK(self):  # 定义 Hero 类的GANK方法…
```

```
a = Hero()  # 对象 a 是 Hero 类的一个实例
b = Hero()  # 对象 b 也是 Hero 类的一个实例
c = Hero()  # 对象 c 也是 Hero 类的一个实例

# 调用类的方法
a.set_info("Batman", 12)
b.set_info("Superman", 13)
c.set_info("Spiderman", 15)
```

继承

继承Hero类

```
49  class Hanxin(Hero):  # Hanxin是Hero的子类，Hero是Hanxin的父类
50      def __init__(self, name, level):
51          super().__init__(name, level)  # 调用父类（Hero类）的构造函数
52
53      def use_skill(self, number):  # 覆盖父类的方法…
61
62      def StealTower(self):  # 扩展出新的方法…
64
```

- 继承：从已有的类派生出新的类（子类）
  - 子类继承父类的数据属性和行为
  - 可以调用父类的函数
  - 可以扩展出新的方法
  - 或者覆盖父类的方法


```
h = Hanxin("韩信", 15)
h.GANK()  # 注意到，Hanxin类中并没有定义GANK函数
h.use_skill(3)
h.StealTower()
```

多态

- 多态：允许对不同的对象进行相同的操作。

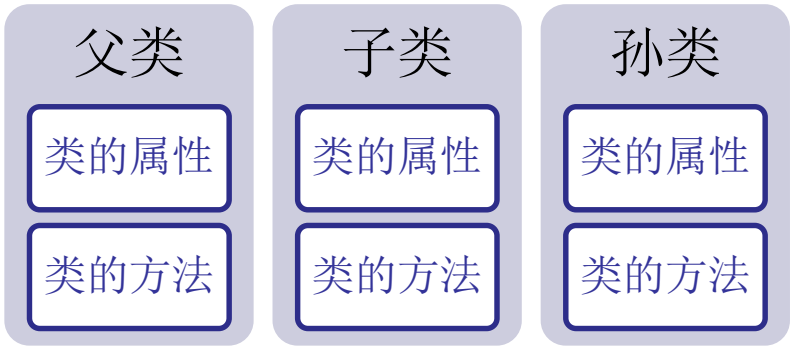
```
def show(hero):
    hero.use_skill(2)
    hero.GANK()


l = Libai("李白", 13)
h = Hanxin("韩信", 4)
b = BaiLongYin("白龙吟", 15, 1188)
show(l)
show(h)
show(b)
```

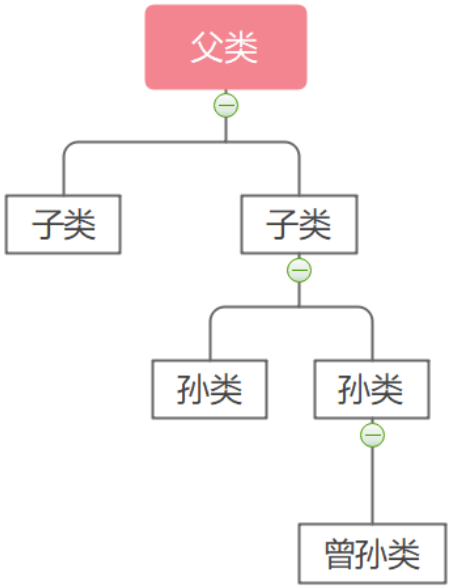
```
13 级的 李白 使用了 2 技能：神来之笔
13 级的 李白 前来支援！
4 级的 韩信 使用了 2 技能：背水一战
4 级的 韩信 前来支援！
15 级的 白龙吟 使用了 2 技能：背水一战
15 级的 白龙吟 前来支援！
```

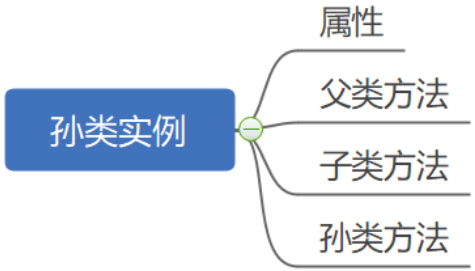
面向对象的三大特性

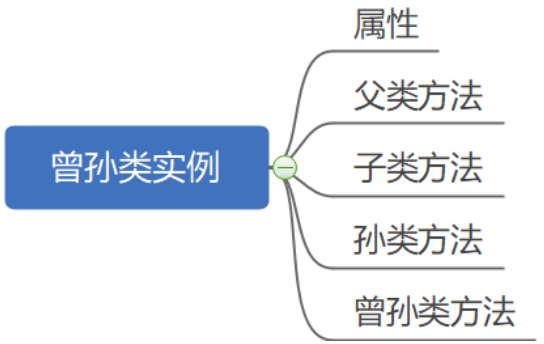
- 封装：隐藏不必要的内部实现细节，对外公开接口。
- 继承：从已有的类派生出新的类（子类），子类继承父类的数据属性和行为，并能根据自己的需求扩展出新的方法或者覆盖父类的方法。
- 多态：允许对不同的对象进行相同的操作。



参见“大纲”







其他语法*

- 异常处理
  - try…except…else
- 迭代器、生成器、修饰器
- 面向对象高级编程
  - 多重继承
  - 定制类
  - @property
  - 元类
- 多线程、多进程、分布式……

第三讲：类与对象、模块

- 类与对象
- 模块
  - 内建模块
  - 第三方库

模块/库/包

- **开箱即用，“batteries included”**
- 模块（module）：一个.py文件就称之为一个模块
- 包（package）：有层次的文件目录结构，可能有n个模块或n个子包
- 库（library）：能完成一定功能、供用户使用的代码集合（抽象概念），在python 中是包和模块的形式。

- 优点：
- 提高了代码的可维护性
- 不必从零开始

调用自定义模块

- 几种import的方法

```
# 三种常见的 import 方法
import L3_inherit  # 直接导入
import L3_inherit as cc  # 导入并且起一个别名
from L3_inherit import Hanxin, Libai  # 选择性导入一部分

# 非常不建议的写法
from L3_inherit import *  # 导入模块中的全部
```

- main 函数
  - 只有当该Python文件直接运行的时候才执行
  - 当该python文件被作为模块import时，main()函数将不会被执行

```
if __name__ == "__main__":
    pass
```

内建模块

- 时间与日期
  - time模块
  - **datatime模块**：重新封装了time模块，提供更多接口

```
import datetime
now = datetime.datetime.now()
print(now)
```

- 中文日期：locale模块
  - 根据计算机用户所使用的语言，所在国家或者地区，以及当地的文化传统所定义的一个软件运行时的语言环境

```
print(now.strftime('%Y-%m-%d %H:%M'))

import locale
locale.setlocale(locale.LC_CTYPE, 'chinese')
print(now.strftime('%Y年%m月%d日%H时%M分'))
```

```
2020-07-19 02:20:38.504325
2020-07-19 02:20
2020年07月19日02时20分
```

内建模块

- 随机数：random模块

```python
import random

print(random.random())  # 生成一个 [0,1] 之间的随机浮点数
print(random.randint(100, 200))  # 生成一个 [a,b] 之间的随机整数

print(random.uniform(3.3, 4.0))  # 产生 [a,b] 之间的随机浮点数，区间可以不是整数
print(random.randrange(0, 1000, 100))  # 生成 [a,b] 之间间隔为整数的随机整数
print(random.choice("abcdefghijklmnopqrstuvwxyz"))  # 从序列中随机选取一个元素
```

- 可以控制随机数种子
- 更复杂的随机数：numpy模块，见第五讲

其他内建模块*

- 操作系统、路径处理相关：os模块，见第四讲
- JSON文件操作：json模块，见第四讲
- 正则表达式：re模块，见第六讲
- 网络请求：requests模块，见第六讲

- 命令行参数： argparse模块
- 日志输出：logging模块
- 解释器相关：sys模块
- 集合对象、队列结构：collections模块
- 操作迭代对象：itertools模块
- ……

- 更多内建模块介绍：常用内建模块

安装第三方库

- 命令行：pip install <要安装的包的名字>

```
问题  输出  调试控制台  终端                                    2: Python Debug Consc

(base) E:\Testfield\summer_python\Lecture_3>pip install easygui
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Requirement already satisfied: easygui in e:\softwares\anaconda\lib\site-packages (0.98.1)

(base) E:\Testfield\summer_python\Lecture_3>
```

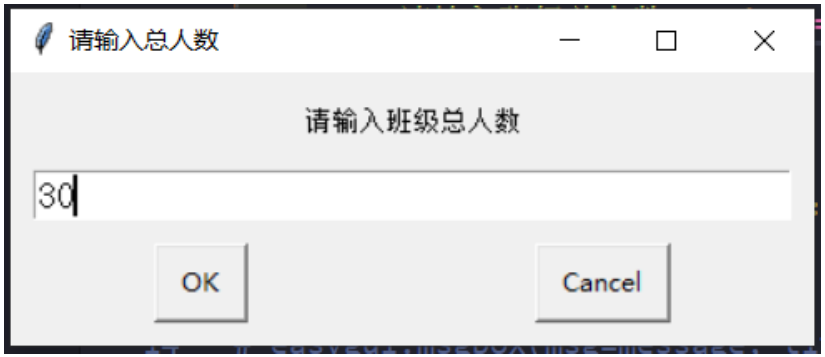
- 包管理软件：Anaconda*

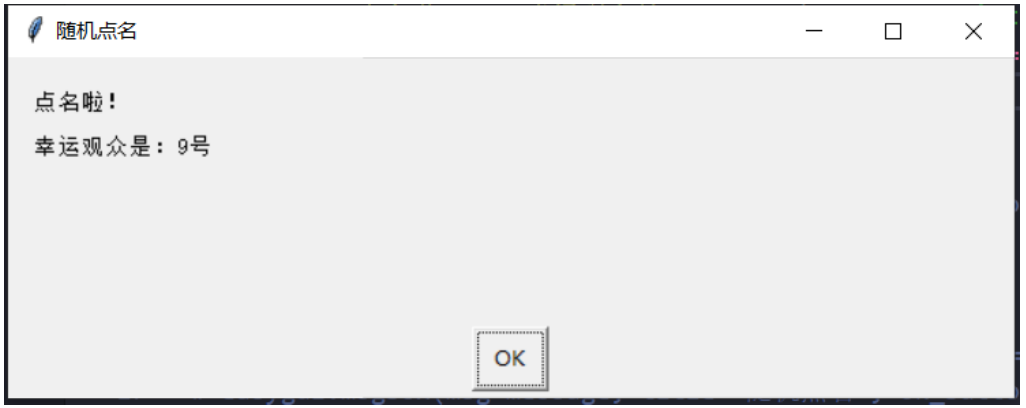
第三方库

- 简单界面：easygui模块

```
import random
import easygui

num = easygui.enterbox(
    msg="请输入班级总人数", title="请输入总人数", default=30, strip=True, image=None, root=None
)
message = "点名啦！\n\n幸运观众是：" + str(random.randint(1, int(num))) + "号"
easygui.msgbox(msg=message, title="随机点名", ok_button="OK", image=None, root=None)
```

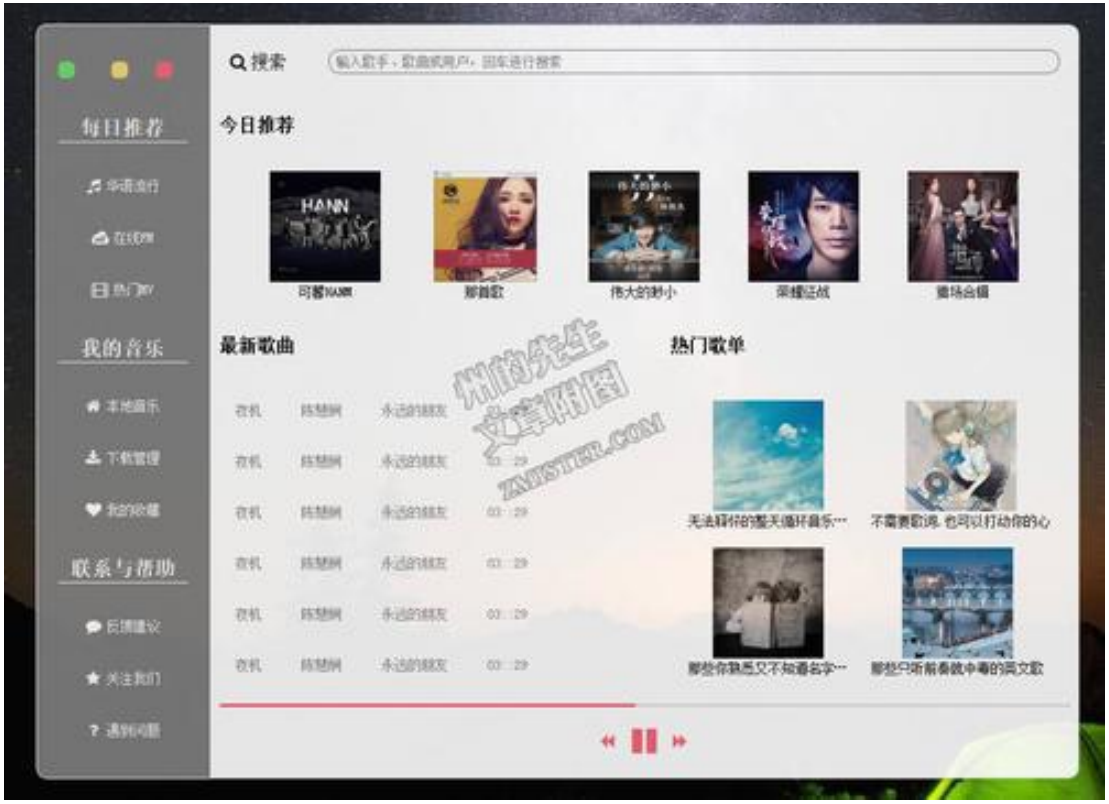




- 更详细的教程：[Python 模块EasyGui详细介绍]

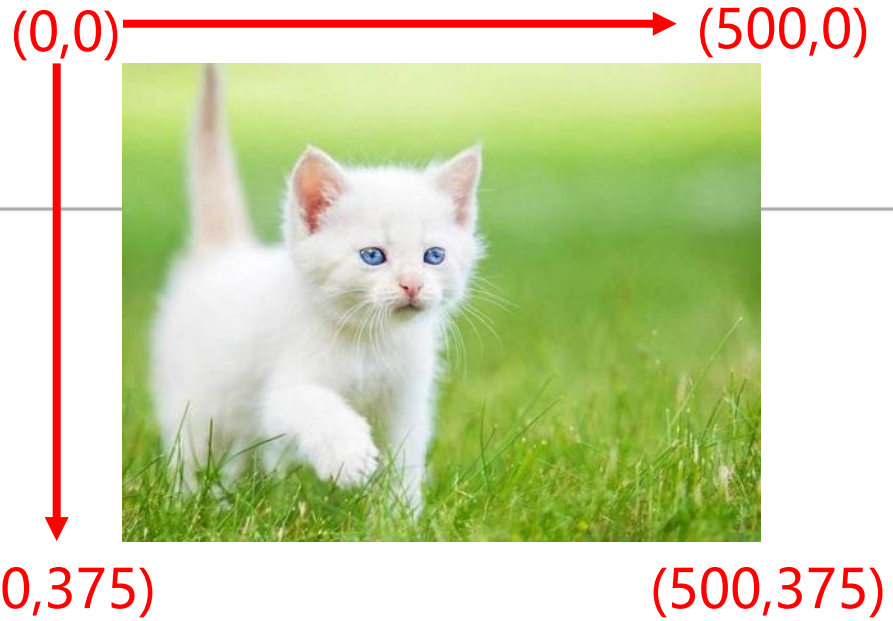
PyQt5*

- 更高级的界面：pyqt5模块



- [Python GUI教程（十六）：在PyQt5中美化和装扮图形界面]

第三方库



- 图像操作：pillow模块

```python
from PIL import Image

im = Image.open("cat.jpg")
box = (0, 0, 300, 375)
region = im.crop(box)  # 裁剪
region.save("cat_croped.jpg")

# 改变尺寸
im = im.resize((800, 375))
im.save("cat_resized.jpg")
```

```python
from PIL import Image, ImageFilter
im = Image.open("cat.jpg")
im2 = im.filter(ImageFilter.GaussianBlur)  # 高斯模糊
im3 = im.filter(ImageFilter.BLUR)  # 普通模糊
im4 = im.filter(ImageFilter.EDGE_ENHANCE)  # 边缘增强
im5 = im.filter(ImageFilter.FIND_EDGES)  # 找到边缘
im6 = im.filter(ImageFilter.EMBOSS)  # 浮雕
im7 = im.filter(ImageFilter.CONTOUR)  # 轮廓
im8 = im.filter(ImageFilter.SHARPEN)  # 锐化
im9 = im.filter(ImageFilter.SMOOTH)  # 平滑
im10 = im.filter(ImageFilter.DETAIL)  # 细节
```

- 调整尺寸、裁剪、滤波、翻转、旋转、调色、粘贴、增加文字、各种格式的图像读写……
- 更详细的教程：官方文档，中文教程

其他第三方库*

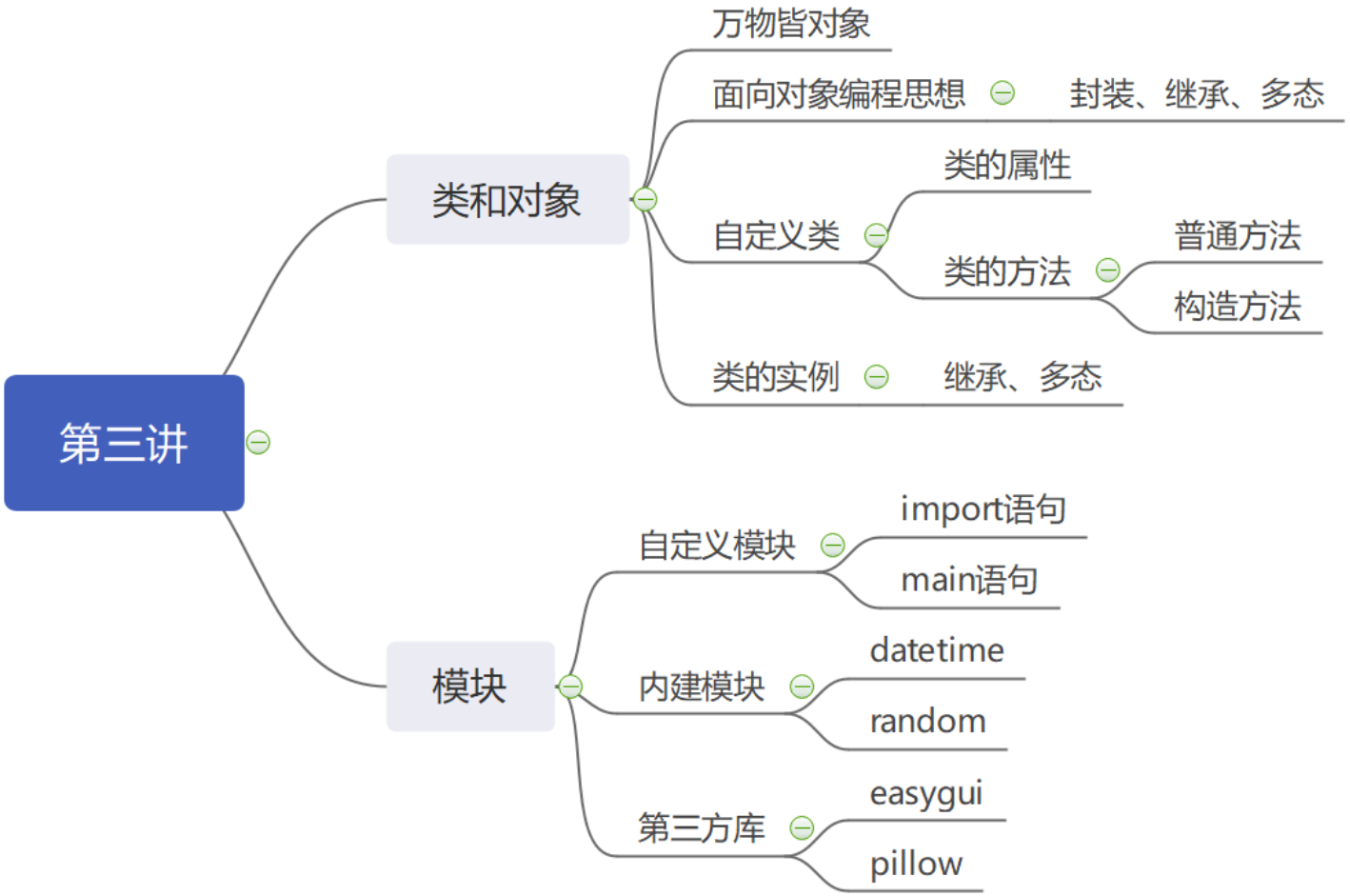
- 操作Word：python-docx模块，见第四讲
- 操作Excel：xlwings模块，见第四讲
- 操作Pdf：PyPDF2模块，见第四讲

- 科学计算：numpy模块，见第五讲
- 数据分析：pandas模块，见第五讲

- 模糊字符串匹配：fuzzywuzzy
- 中文分词工具：jieba
- 开源计算机视觉库：OpenCV
- 深度学习框架：pytorch、tensorflow、keras……
- ……数不胜数

- 哪些 Python 库让你相见恨晚?

第三讲总结


第三讲
类和对象
万物皆对象
面向对象编程思想
封装、继承、多态
自定义类
类的属性
类的方法
普通方法
构造方法
类的实例
继承、多态
模块
自定义模块
import语句
main语句
内建模块
datetime
random
第三方库
easygui
pillow

课后练习

1. **(基本要求)** 自己设计一系列类并且调用，要求用到构造函数、继承、多态。

2. **(基本要求)** 写一个可以随机选择食堂的程序，要求带有界面输出。

3. **(复习，选作)** 注册Leetcode并做一些“简单”题目：
https://leetcode-cn.com/problems/can-make-arithmetic-progression-from-sequence/
https://leetcode-cn.com/problems/maximum-product-of-two-elements-in-an-array/
https://leetcode-cn.com/problems/robot-return-to-origin/