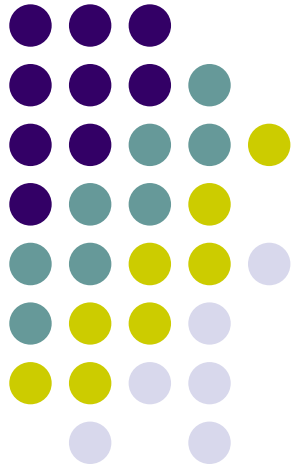
数据分析理论与Python实战

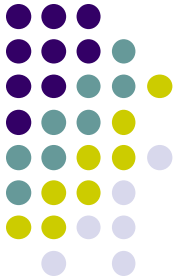
第二章 Python——从了解Python开始



目录

- Python发展史
- Python及相关包的安装
- Python基础知识
- 重要的Python库
- Jupyter

Python发展史



1989年圣诞节：荷兰数学家、计算机学家Guido von Rossum发明Python

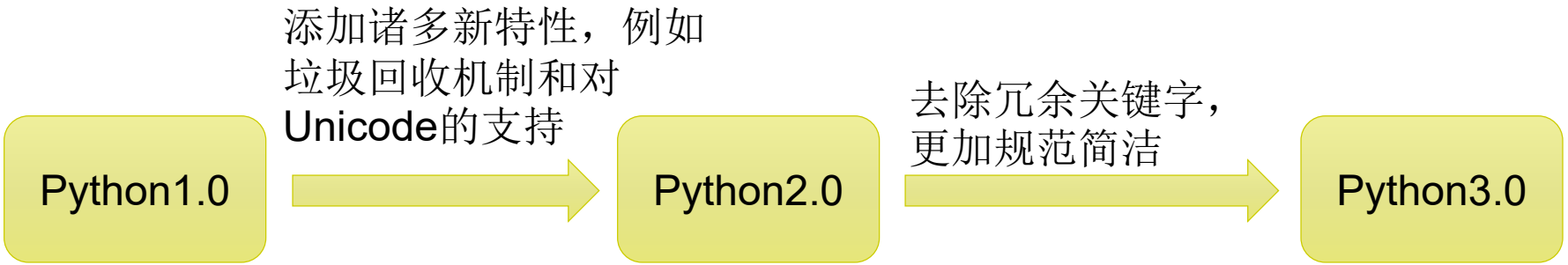
1991年：Python第一个公开发行版问世

2000年10月：Python2.0版本发行

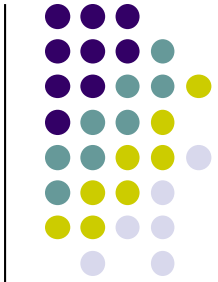
2008年12月：Python3.0版本发行

Python发展史

- Python版本进化过程

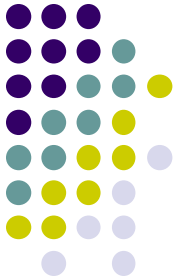


Python及相关包的安装



- Python的安装
  - Windows环境
    - 在官网下载相应安装程序进行安装
  - Mac环境
    - 在官网下载相应安装程序进行安装
    - 使用homebrew进行安装

Python及相关包的安装



- 相关包的安装
  - 使用pip进行安装
    - Pandas
    - Scikit-learn
    - Matplotlib

Python及相关包的安装

- 使用科学计算发行版Python进行快速安装
  - 一般会包含一个标准版本的python和多个科学计算相关的包
  - 流行的科学计算发行版Python
    - Anaconda
    - WinPython

Python基础知识例：求斐波那契数列

单行注释

多行注释

引入包

函数

函数头

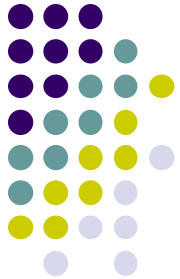
函数体

函数调用

调用os包中的函数

```
1:  #Fibonacci sequence
2:  '''
3:  斐波那契数列
4:  输入：项数n
5:  输出：前n项
6:  '''
7:  import os
8:
9:  def fibo(num):
10:     numbers=[1,1]
11:    for i in range(num-2):
12:        numbers.append(numbers[i]+numbers[i+1])
13:    return numbers
14:
15: answer=fibo(10)
16: print(answer)
17:
18: if not os.path.exists('result'):
19:     os.mkdir('result')
20:
21: file=open('result/fibo.txt','w')
22:
23: for num in answer:
24:     file.write(str(num)+' ')
25:
26: file.close()
```

Python基础知识



- 缩进很重要
  - 缩进符决定了程序的结构

```
if not os.path.exists('result'):
    os.mkdir('result')
```

分支结构中，分支体需要缩进

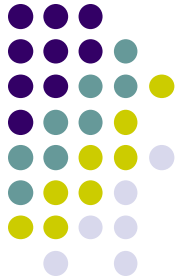
```
for num in answer:
    file.write(str(num)+' ')
```

循环结构中，循环体需要缩进

```
def fibo(num):
    numbers=[1,1]
    for i in range(num-2):
        numbers.append(numbers[i]+numbers[i+1])
    return numbers
```

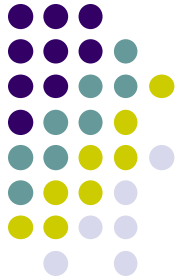
函数体需要缩进

Python基础知识



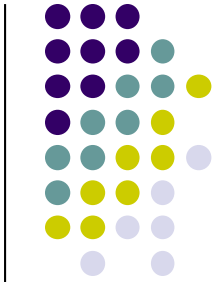
- 模块化的系统
  - Python拥有功能丰富的标准库和强大的第三方库支持
  - 标准库
    - os（提供操作系统各类接口）
    - time（提供对日期和时间的处理）
  - 第三方库
    - scipy（科学计算）
    - scikit-learn（机器学习库）

重要的Python库



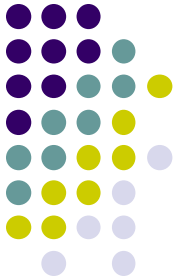
- Pandas
  - 构建在Numpy之上的高性能数据分析库
  - 对数据进行排序、分组、归并等操作和求和、求极值、求标准差、协方差矩阵计算等统计计算
- Scikit-learn
  - 构建在Numpy、Scipy和matplotlib上的机器学习库
  - 包括多种分类、回归、聚类、降维、模型选择和预处理算法与方法（例如支持向量机、最近邻、朴素贝叶斯、LDA、特征选择、k-means、主成分分析、网格搜索、特征提取等等）

重要的Python库



- Matplotlib
  - 一个绘图库
  - 可绘制直方图、折线图、饼图、散点图、函数图像等2D、3D图形，甚至是动画
- 其他
  - Numpy（科学计算库）
  - Scipy（科学计算库）
  - Scrapy（网络爬虫库）
  - NLTK（自然语言处理库）
  - Statsmodels（统计学计算库）

Jupyter



- 交互式的数据科学与科学计算开发环境
  - 支持Python、R、Scala等在内的超过40多种编程语言
  - Jupyter notebook
    - 基于web的python编辑器
    - 使用markdown语言将样式丰富的文字添加到notebook中，实现代码、运行结果和文字的穿插展示

Jupyter使用样例

双扭线的绘制

参考百度百科：双扭线

双扭线是什么?

- 双纽线，也称伯努利双纽线
- 设定线段AB长度为2a，若动点M满足MA*MB=a^2，那么M的轨迹称为双纽线
- 双扭线的极坐标方程为 $\rho = a^2 \cos 2\theta$

利用matplotlib绘制双扭线

相比平面直角坐标系中的函数图像绘制，在极坐标系中绘制函数图像需要在建立Axe时指定投影（projection）参数为极坐标（polar）。首先我们根据双扭线的极坐标方程生成了两组数据

```
theta_list = np.arange(0, 2*np.pi, 0.01)
r = [2*np.cos(2*theta) for theta in theta_list]
```

然后，我们建立一个投影为极坐标的Axe

```
axe = plt.subplot(projection='polar')
```

接下来，使用Axe.plot()函数生成函数曲线，为了使图形更加美观，删除了r轴上的所有tick

```
axe.plot(theta_list, r)
axe.set_rticks([])
```

最后，使用pyplot.show()函数展示图形

```
plt.show()
```

完整代码和运行结果如下：

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
theta_list = np.arange(0, 2*np.pi, 0.01)
r = [2*np.cos(2*theta) for theta in theta_list]
axe = plt.subplot(projection='polar')
axe.plot(theta_list, r)
axe.set_rticks([])
plt.show()
```