PYTHON数据科学系列课程（一）
课程介绍和NUMPY入门

东南大学 学习科学研究中心 儿童发展与教育研究所

夏小俊

http://www.seuct.com

内容提要

1.1 课程介绍

Python的设计哲学

- 优雅、明确、简练
- Life is short，you need Python

Python的设计目的

- 符合数据分析与数据科学项目的需要

Python的受欢迎程度

- 已成为数据分析和数据科学领域最受欢迎的语言

培养目标

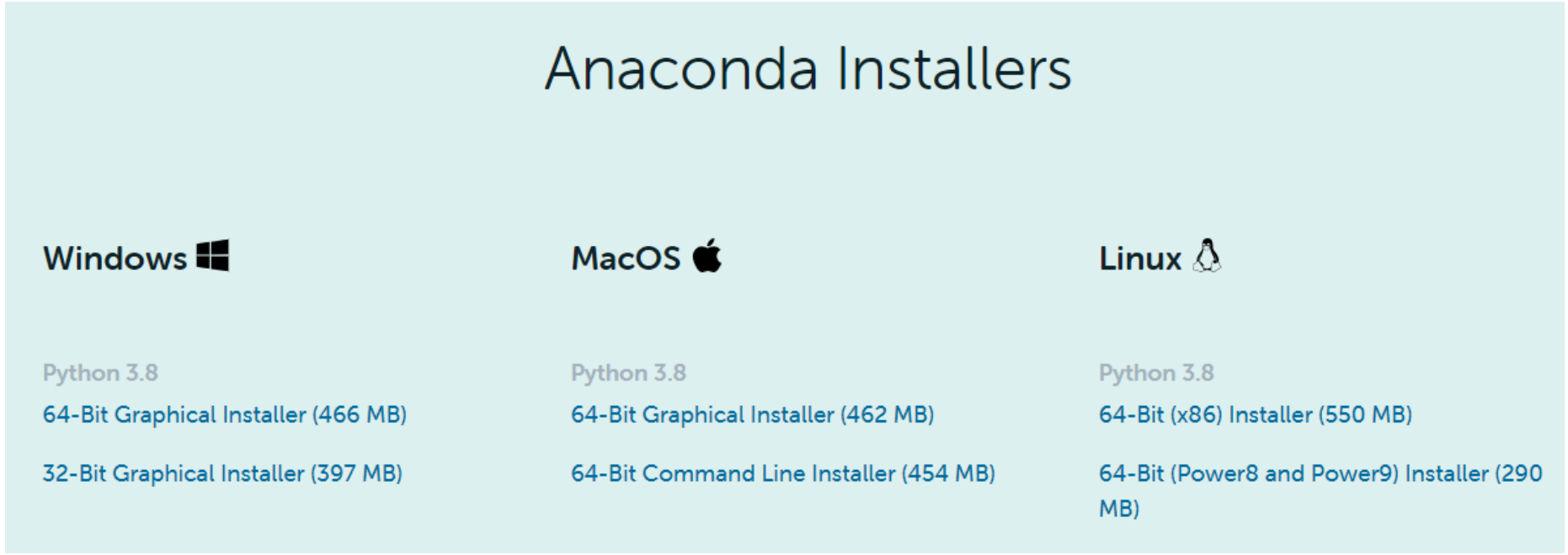
- 本课程以数据分析/数据科学为目的与导向，以Python语言为工具，帮助学生掌握基于Python的数据获取、加工、可视化、统计分析等技能。
- 课程主要内容以数据科学相关的Python语言及第三方扩展库（numpy/pandas/matplotlib/requests等）的基础知识、思路、方法、经验和技巧，打通了从Python到传统数据分析再到现代数据科学的完整通道。在学生已有Python编程语言的基础之上，着重关注对学生数据科学的思维和方法培养。
- 课程建议采用基于项目式学习(Project Based Learning)和合作式学习（CPS）的方法，以教学或生活中的案例为引导，通过有真实性、挑战性、可行性的课题让学生在实践中持续学习，关注学生批判性思维能力、解决问题的能力的培养。**课程实施当中，师生可以充分讨论，进行评论、反思和修正，最终完成完善自己的项目。**

1.2 关于NUMPY

- NumPy 是一个 Python 包，代表 “Numeric Python” ，是一个由多维数组对象和用于处理数组的例程集合组成的库，由Travis Oliphant等人在2005年创建。
- NumPy是Python中用于数据分析、机器学习、科学计算的重要软件包，极大地简化了向量和矩阵的操作及处理。
- 大量的Python数据处理包都将NumPy作为其基础架构的核心部分（例如scikit-learn、SciPy、Pandas和tensorflow）。
- NumPy官网 https://numpy.org/ 中文官网 https://www.numpy.org.cn

1.2 关于NUMPY

- 单独安装NumPy：pip install numpy 或者conda install numpy
- 安装集成环境Anaconda，自带数据分析常用等上千个包。



1.3 创建NDARRAY对象

- 数组就是有序的元素序列，Python内置的列表、元组也可以看成是数组，但类似列表的底层实现比较复杂，存储和运算效率都比较低。
- NumPy 的大部分代码都是用 C 语言写的，其底层算法在设计时（统一的数据类型、连续的存储方式等）就有着极优异的性能，所以使得 NumPy 比纯 Python 代码高效得多。
- NumPy 的三个关键知识点：数组类型的创建、数据层的索引切片、数组运算。
- 虽然大多数的数据分析工作并不会直接操作 Numpy 对象，但是深谙面向数组的编程方式和逻辑能力是培养数据分析思维的关键。

1.3 创建NDARRAY对象

In[1]:
```
import numpy as np
```

In[2]:
```
MyArray1 = np.arange(1,20)
MyArray1
```

Out[2]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19])
```

In[3]:
```
range(1,10,2)
```

Out[3]:
```
range(1, 10, 2)
```

1.3 创建NDARRAY对象

In[4]:
```
list(range(1,10,2))
```
Out[4]: [1, 3, 5, 7, 9]

In[5]:
```
np.arange(1,10,2)
```
Out[5]: array([1, 3, 5, 7, 9])

In[6]:
```
MyArray2=np.array([1,2,3,4,3,5])
MyArray2
```
Out[6]: array([1, 2, 3, 4, 3, 5])

1.3 创建NDARRAY对象

In[7]:

```
np.array(range(1,10,2))
```

Out[7]:

```
array([1, 3, 5, 7, 9])
```

In[8]:

```
MyArray3=np.zeros((5,5))
MyArray3
```

Out[8]:

```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

1.3 创建NDARRAY对象

In[9]:

```
MyArray4=np.ones((5,5))
MyArray4
```

Out[9]:

```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

In[10]:

```
np.full((3,5),2) #试试empty
```

Out[10]:

```
array([[2, 2, 2, 2, 2],
       [2, 2, 2, 2, 2],
       [2, 2, 2, 2, 2]])
```

1.3 创建NDARRAY对象

In[11]:

```
rand=np.random.RandomState(30)

MyArray5=rand.randint(0,100,[3,5])
MyArray5
```

Out[11]:

```
array([[37, 37, 45, 45, 12],
       [23,  2, 53, 17, 46],
       [ 3, 41,  7, 65, 49]])
```

1.3 创建NDARRAY对象

In[12]:

```
x = np.linspace(10,20,5)
print(x)

x = np.linspace(10,20,  5, endpoint =  False)
print(x)
```

Out[12]:

```
[10.  12.5 15.  17.5 20. ]
[10. 12. 14. 16. 18.]
```

1.3 创建NDARRAY对象

In[13]:

```
import numpy as np
a = np.logspace(1,10,num = 10, base = 2)
print(a)
```

Out[13]: [2. 4. 8. 16. 32. 64. 128. 256. 512. 1024.]

In[14]:

```
np.eye(3) #np.identity(3)
```

1.3 创建NDARRAY对象

asarray : asarray(a, dtype = None, order = None)
功能:用于从序列当中创建数组

x = [(1,2,3),(4,5,6)]
a = np.asarray(x)

frombuffer: frombuffer(buffer, dtype = float, count = -1, offset = 0)
功能:用于从缓冲区当中读取数据并创建数组

fromiter:fromiter(iterable, dtype, count = -1)
功能:从可迭代对象当中读取数据并创建数组

1.4 NDARRAY对象数据类型

1. bool_ 存储为一个字节的布尔值（真或假）
2. int_ 默认整数，相当于 C 的long，通常为int32或int64
3. intc 相当于 C 的int，通常为int32或int64
4. intp 用于索引的整数，相当于 C 的size_t，通常为int32或int64
5. int8 字节（-128 ~ 127）
6. int16 16 位整数（-32768 ~ 32767）
7. int32 32 位整数（-2147483648 ~ 2147483647）
8. int64 64 位整数（-9223372036854775808 ~ 9223372036854775807）
9. uint8 8 位无符号整数（0 ~ 255）
10. uint16 16 位无符号整数（0 ~ 65535）
11. uint32 32 位无符号整数（0 ~ 4294967295）
12. uint64 64 位无符号整数（0 ~ 18446744073709551615）
13. float_ float64的简写
14. float16 半精度浮点：符号位，5 位指数，10 位尾数
15. float32 单精度浮点：符号位，8 位指数，23 位尾数
16. float64 双精度浮点：符号位，11 位指数，52 位尾数
17. complex_ complex128的简写
18. complex64 复数，由两个 32 位浮点表示（实部和虚部）
19. complex128 复数，由两个 64 位浮点表示（实部和虚部）

numpy 的数值类型是 dtype 对象的实例，并对应唯一的字符，包括 np.bool_，np.int32，np.float32等等，也可以用双引号直接加类型名表示。

1.4 NDARRAY对象数据类型

每个内建类型都另有一个唯一定义它的字符代码，可以配合类型长度（字节数）和字节序来书写，注意添加引号。

字符	对应类型
b	布尔型
i	(有符号) 整型
u	无符号整型 integer
f	浮点型
c	复数浮点型
m	timedelta（时间间隔）
M	datetime（日期时间）
O	(Python) 对象
S, a	(byte-)字符串
U	Unicode
V	原始数据 (void)

1.4 NDARRAY对象数据类型

In[1]:
```
import numpy as np
np.zeros(10,dtype="int16")
```
Out[1]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int16)
```
In[2]:
```
np.zeros(10,dtype="float")
```
Out[2]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```
In[3]:
```
a1=np.array([1,2,3,None])
```
Out[3]:
```
array([1, 2, 3, None], dtype=object)
```
In[4]:
```
a1=np.array([1,2,3,None,np.nan])
```
Out[4]:
```
array([1, 2, 3, None, nan], dtype=object)
```

1.4 NDARRAY对象数据类型

In[5]:
```
arr = np.array([1,2,3,4,5])
float_arr = arr.astype(np.float64) #切忌修改dtype
```

Out[5]:
```
array([1., 2., 3., 4., 5.])
```

In[6]:
```
numeric_strings = np.array(['1.2','2.3','3.2141'])
numeric_strings.astype(float)
```

Out[6]:
```
array([1.2   , 2.3   , 3.2141])
```

In[7]:
```
numeric_strings = np.array([1.2,2.3,3.2141])
numeric_strings.astype('<U6')
```

Out[7]:
```
array(['1.2', '2.3', '3.2141'], dtype='<U6')
```

1.5 切片/读取

In[1]:
```
myArray=np.arange(1,10)
```

Out[1]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In[2]:
```
myArray[0]
```

Out[2]: 1

In[3]:
```
myArray[-1]
```

Out[3]: 9

1.5 切片/读取

In[4]:

```
import numpy as np
myArray=np.array(range(0,10))
print("myArray=",myArray)
print("myArray[1:9:2]=",myArray[1:9:2])
print("myArray[:9:2]=",myArray[:9:2])
print("myArray[::2]=",myArray[::2])
print("myArray[::]=",myArray[::])
print("myArray[:8:]=",myArray[:8:])
print("myArray[:8]=",myArray[0:8])
print("myArray[4::]=",myArray[4::])
print("myArray[9:1:-2]=",myArray[9:1:-2])
print("myArray[::-2]=",myArray[::-2])
print("myArray[[2,5,6]]=",myArray[[2,5,6]])
print("myArray[myArray>5]=",myArray[myArray>5])
```

1.5 切片/读取

Out[4]:
```
myArray= [0 1 2 3 4 5 6 7 8 9]
myArray[1:9:2]= [1 3 5 7]
myArray[:9:2]= [0 2 4 6 8]
myArray[::2]= [0 2 4 6 8]
myArray[::]= [0 1 2 3 4 5 6 7 8 9]
myArray[:8:]= [0 1 2 3 4 5 6 7]
myArray[:8]= [0 1 2 3 4 5 6 7]
myArray[4::]= [4 5 6 7 8 9]
myArray[9:1:-2]= [9 7 5 3]
myArray[::-2]= [9 7 5 3 1]
myArray[[2,5,6]]= [2 5 6] #花式索引
myArray[myArray>5]= [6 7 8 9] #布尔索引
```

1.5 切片/读取

In[5]:
```
myArray[1,3,6]
```

```
IndexError                                Traceback (most recent call
last)
<ipython-input-30-13b1cd8a6af6> in <module>()
      1
----> 2 myArray[1,3,6]
      3
IndexError: too many indices for array
```

In[6]:
```
myArray[[1,3,6]]
```

Out[6]: array([2, 4, 7])

1.5 切片/读取

In[7]:

```
import numpy as np
a = np.array([[1,2,3],[3,4,5],[4,5,6]])
a[...,1:3] #... Ellipsis
```

Out[7]: array([[2, 3], [4, 5], [5, 6]])

1.5 切片/读取

In[8]:
```
myArray2 = np.arange(1,21).reshape(5,4)
myArray2[[2,4],3] #花式索引,还可以和:, ...等结合
```

Out[8]: array([12, 20])

In[9]:
```
x=[2,4]
myArray2[x,3]
```

Out[9]: array([12, 20])

In[10]:
```
x=[2,4];y=[0,0]
myArray2[x,y] #整数数组索引
```

Out[10]: array([9, 17])

1.6 赋值与拷贝

In[1]:

```
import numpy as np
myArray1=np.array(range(0,10))
myArray2=myArray1
myArray2[1]=100 #切片和高级索引可能产生类似情况
myArray1
```

Out[1]: array([0, 100, 2, 3, 4, 5, 6, 7, 8, 9])

In[2]:

```
import numpy as np
myArray1=np.array(range(0,10))
myArray2=myArray1.copy()
myArray2[1]=200
myArray1
```

Out[2]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

谢谢大家！