需要的python程序基础

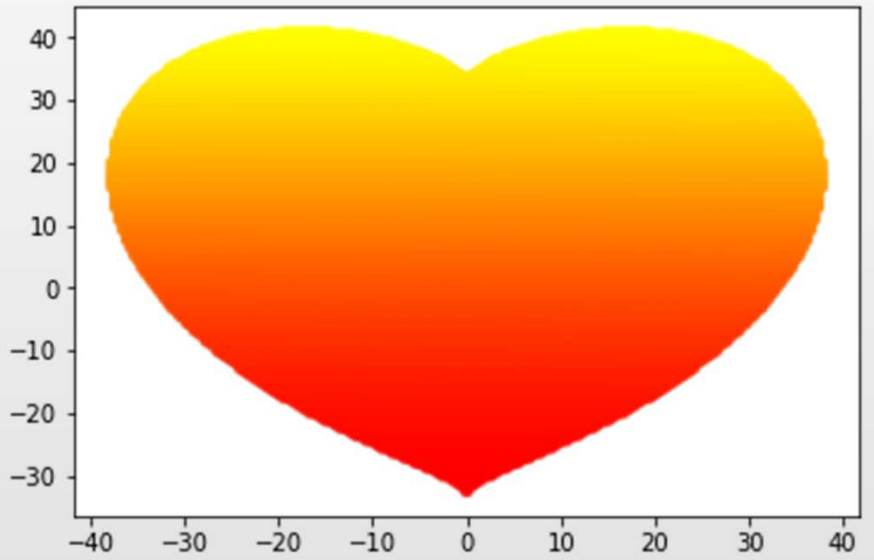
1.Python程序基础

需要掌握——

- 变量
- 选择结构
- 循环结构
- 函数及调用
- 文件读写

实验1：一个简单的python程序

```
# -*- coding: UTF-8 -*-
import numpy as np
import matplotlib.pyplot as plt

def draw(FillStyle):
    x_coords = np.linspace(-100, 100, 500)
    y_coords = np.linspace(-100, 100, 500)
    points = []

    for y in y_coords:
        for x in x_coords:
            if ((x*0.03)**2+(y*0.03)**2-1)**3-(x*0.03)**2*(y*0.03)**3 <= 0: # 引
用公式
                points.append({"x": x, "y": y})

    heart_x = list(map(lambda point: point["x"], points))
    heart_y = list(map(lambda point: point["y"], points))

    if FillStyle==1:
        plt.scatter(heart_x, heart_y, s=10, alpha=0.5)
    else:
        plt.scatter(heart_x, heart_y, s=10, alpha=0.5, c=range(len(heart_x)),
cmap='autumn')
    plt.show()

#主过程
fStyle=2
draw(fStyle)
```



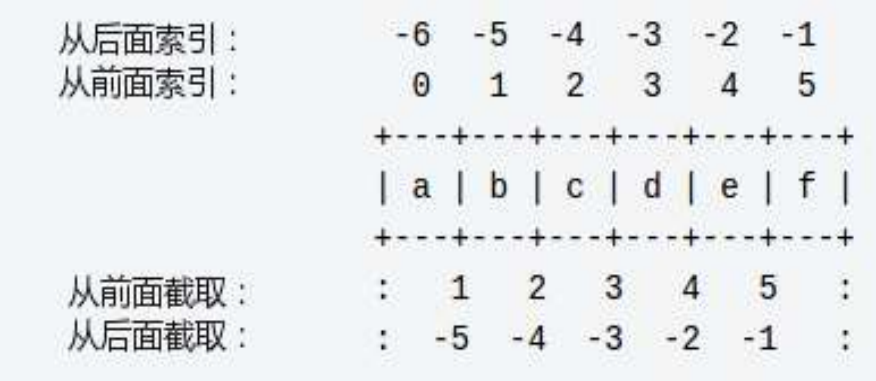
Python数据处理

一.python数据处理

- Python3 中有六个标准的数据类型：
  - Number（数字）
  - String（字符串）
  - List（列表）
  - Tuple（元组）
  - Set（集合）
  - Dictionary（字典）

1、String（字符串）类型

- Python中的字符串用单引号 ' 或双引号 " 括起来，同时 使用反斜杠 \ 转义特殊字符。
- 字符串的截取的语法格式如下：
  - 变量[头下标:尾下标]
- 访问方法：



实验2.字符串操作示例

```
#字符型
str = 'PyString'
print (str[1:3])   # 第二、三个字符
print (str[-3:-1]) # 倒数第三个、倒数第二个字符
print (str[2:-1]) # 正数第三个到倒数第二个字符
print (str[2:]) # 第三个字符开始的所有字符
```

2、List（列表）类型

- List（列表） 是写在方括号 [] 之间、用逗号分隔开的元素列表。 列表是 Python 中使用最频繁的数据类型。可以包含数字，字符串 甚至可以包含列表（所谓嵌套）。

  list = ['a', 56 , 1.13, 'picture',[7,8,9]]

- 列表截取的语法格式也是：
  - 变量[头下标:尾下标]

实验3：列表数据的遍历

```
lis= ['dog','cat','cow','duck','pig']
```

方法1——直接遍历

```
for item in lis:
        print(item)
```

方法2——使用下标遍历

```
for i  in range(len(lis)):
        print(lis[i])
```

3、Tuple（元组）类型

元组（tuple）与列表类似，但元组的元素不能修改。元组写在小括号（）里，元素之间用逗号隔开。另外，元组中的元素类型也可以不相同。

元组与字符串类似，可以被索引且下标索引从0开始，-1 为从末尾开始的位置。

例如：

tuple = (‘SpiderMan', 786 , 2.23, ‘Homecoming', 70.2)

```
tuple = ( 'SpiderMan',2017 ,33.4, 'Homecoming', 14 )
tinytuple = (16, 'Marvel')
print (tuple)          # 输出完整元组
print (tuple[3:4])   # 输出从第二个元素开始到第三个元素
print (tuple + tinytuple) # 连接元组
```

实验4：修改元组中的改变元素

```
#修改元组中的list类型数据项
tuple = (['dog','cat','cow','duck','pig'], 2017 ,33.4,'ha')
tuple[0][0]=1
tuple[0][1]='ant‘
print(tuple)
```

4、Dictionary（字典）

字典是一种可变容器模型，且可存储任意类型对象。字典的每个键值**key/value** 对用冒号**:** 分割，每个键值对之间用逗 号**,** 分割，整个字典包括在**{}** 中 ,格式如下：

d = {key1 : value1, key2 : value2 }

例如：

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'};

访问的时候需要使用key 进行索引，如下：

print(dict['Name']);

实验5：定义字典并添加字典元素

```
#字典
dict = {'type':['dog','cat','cow','duck','pig'], 'Age':[1,2,3,4,5]};
print("type: ", dict['type'])
print("Age: ", dict['Age'])

#直接添加字典元素
dict['speed']=110
dict
```

5、Set（集合）类型

- set 具有一系列元素，list 类似。但是set的元素是不重复且无序的。
- 创建set的方式是调用 set()或使用大括号 { }，并传入一个 list，list的元素将作为set的元素。
- 元素顺序和原始list的顺序有可能不同，因为set内部存储是无序的。
- 可以使用in方法判断元素是否在集合内。
- 创建一个空集合必须用set()不能用{ }，因为{ }是创建一个空字典。

实验6：定义字典并添加字典元素

```
#空集合
var = set()
print(var,type(var))    #显示集合内容和类型

#具有数据的集合
var = {'dog','cat','cow','duck','pig'}
print(var,type(var))    #显示集合内容和类型

#判断元素在集合内
result = 'mouse' in var
print(result)
```

练习：

遍历anml ={'紫貂','松貂','青鼬','狼獾'}集合.

并观察遍历的结果。

- 机器学习的本质是数据处理，及在此基础上的算法运行。

Python文件读取

一、Python读取文件

1.Python打开文件：

- f = open(‘newfile.txt’,‘r’)，包括r、w、a等文件操作方 式。

2.关闭文件使用：

- f.close()

然而，当文件读写产生IOError时，后面的f.close()就不会调用。

为了确保关闭文件，Python引入了with语句，隐含调用f.close()方法。

使用方法：

- with open(filename) as file:

使用了with语句，能获得更好的异常处理。

实验7：定义字典并添加字典元素

运行如下代码，参数'a'表示添加数据，不清除原数据：

```
with open('INFO.txt','a') as f:
    f.write("紫貂,松貂,青鼬,狼獾.\n")
```

- 将上面的程序运行3次，查看INFO.txt文件内容。

读文件的四种方法

- 读取整个文件：f.read()
- 逐行读取数据：f.readline()
- 读当前位置后的所有行，默认位置为文件头：f.readlines()
- 使用迭代器循环读读取当前位置全部行:for item in f

实验8：读文件的练习

```
#1.读取整个文件
with open("INFO.txt") as f:      # 默认模式为‘r’，只读模式
    print( f.read(5))      # 读5个字符
    print('======')
    contents = f.read()    # 从当前位置，读文件全部内容
    print(contents)
```

```
#2.逐行读取数据可以用readline()函数
with open('INFO.txt') as f:
        line1 = f.readline()  # 读取第一行数据（此时已经指向第一行末尾
)
        print(line1)
        print('    ----------------------------------')
        print(line1.strip())
```

#3.读当前位置后的所有行

```
with open('INFO.txt') as f:
    lines = f.readlines()  #为列表，每个元素对应一行
print(lines)              #每一行数据都包含了换行符

for line in lines:
    print(line.rstrip())   #使用rstrip()处理空格
```

```
#4.使用迭代器循环读文件
with open('dataH.txt') as f:
    for lineData in f:
        print(lineData.rstrip()) # 去掉每行末尾的换行符
```

写文件

- 写数据到文件，在打开文件时使用写数据模式‘w’或‘a’, 分别表示改写和添加。
- 使用f.write函数，将内容写入文件。

课堂练习1

打开Jupyter Notebook，建立File.ipynb文件。读文件“fruit_data_with_colors.txt”前10行的数据，并显示；然后把数据写入ften.txt文件。

Pandas文件读取

一、pandas文件读取

Pandas模块提供了特有的文件读取函数，最常用的是处理csv文件的read_csv 函数，其他还有read_table函数等。

- read_csv函数：能从文件、URL、文件对象中加载带有分隔符的数据，默认分隔符是逗号。常用的txt文件和csv文件都可以读取。

实验9：**Pandas存取文件**

```
import pandas as pd
data1 = pd.read_csv('dataH.txt')
print(data1)
print('--   - ---  ------- --  ------')
data2 = pd.read_csv('dataH.txt',sep = '    ',  encoding
= 'utf-8') #指明分隔符
print(data2)
```

使用pandas的to_csv 可以写入文件。

```
data1.to_csv("HW1.csv")
data2.to_csv("HW2.csv")
```

NumPy文件读取

一、Numpy存取文件

使用numpy也能非常方便地存取文件，主要包括下面三组函数：

1.tofile和fromfile()

存取二进制文件

2.load()和save()

存取NumPy专用的二进制格式文件

3.savetxt()和loadtxt()

最为常用，可以存取文本文件，也可以访问csv文件。

格式：np.loadtxt(fname, dtype=, comments='#', delimiter=None, converters=None, skiprows=0, usecols=None, unpack=False, ndmin=0, encoding='bytes’)

常用参数解析——

fname : 文件、字符串或产生器，可以是.gz或.bz2的压缩文件。

dtype : 数据类型，可选 。

delimiter : 分割字符串，默认是任何空格 。

usecols：选取数据的列。

需要注意：np.savetxt() np.loadtxt()只能存取一维和二维数组。

实验10：Numpy存取文件

```
import numpy as np
# 采用字符串数组读取文件
tmp = np.loadtxt("dataH.txt", dtype=np.str, delimiter="  ")
print(tmp)
print("----分隔线----------")
tmp1 = np.loadtxt("dataH.txt",dtype=np.str,usecols=(1,2))
print(tmp1)
```

使用numpy的savetxt，可以把数据写入文件

```
x=[1,2,3]
y=[4,5,6]
z=[7,8,9]
np.savetxt('XYZ.txt', (x,y,z))
```