day5

一、今日内容

a. 单例模式
b. 异常与模块
c. 人工智能简介
d. 人工智能数据
    i. 数据来源
    ii. 数据格式
    iii. 数据类型
    iv. 数据分割
e. K近邻算法 --- 初中数学知识

二、复习

a. 面向对象
    i. 面向过程与面向对象区别
    ii. 类与对象
        1. 类： 类型，泛指一系列的物品
        2. 对象： 特指，特指单一物体，类的具体实例化
    iii. 类的定义
        1. 新式类

```
class 类名(object):
    类成员
    code
object:  python顶级的父类
self:对象本身
```

    iv. 对象定义： 对象名 = 类名()
    v. 类与对象之间的关系：
        1. 类是创建对象的模板
    vi. 魔法方法：
        1. __init__(): 不是创建对象的方法,而是初始化对象的方法
        2. __str__(): 返回一个字符串
        3. __del__(); 在回收资源的时候使用
b. 继承
    i. 继承的格式： class Chirdren(Father): pass

ii. 多继承： 一个子类，继承多个父类

1. __mro__：方法

iii. 子类重写父类的同名方法

iv. 子类调用父类的同名方法： 父类名.方法名()

v. 隔代继承

vi. 私有属性与私有方法：__属性名/ 方法

c. 多态

i. 函数的多种形态：一个函数通过传递不同的参数，最终导致显示的结果不一样。

ii. 鸭子模型

d. 人工智能环境的安装

i. pip install sklearn

ii. pip install pandas

iii. pip install numpy

iv. pip install jieba

v. python -m pip install --upgrade pip

三、单例模式

3.1 概念

单例是一种设计模式： 一个类只能创建出一个对象。应用场景：回收站， 任务管理器。

3.2 __new__方法

功能: 魔法方法, 创建一个对象，返回创建好的对象的的地址。

```
# target: 创建一个单例模式
class Single(object):

    # 定义一个类属性用来记录
    __instance = None

    # 定义一个 __new__方法
    def __new__(cls, *args, **kwargs):
        # __new__ 功能：创建一个对象,在__init__函数之前执行。
        print("this is __new__ function")

        if cls.__instance is None:
            # 如何去创建一个对象 执行了父类的__new__去创建一个对象
            cls.__instance = object.__new__(cls)

        return cls.__instance

```

```
    def __init__(self):

        print("this is __init__ function")


s1 = Single()
s2 = Single()

# 证明S1, S2是同一个对象, id()
print("id(s1) : ", id(s1))
print("id(s2) : ", id(s2))
```

四、模块

4.1 概念

python本身也是支持多文件编程的，每一个单独的文件，都可以看作是一个模块。

4.2 模块的使用

```
# 需要在当前文件中 day5_demo文件中,使用mio模块中的add函数

# 关键字在 import 导入模块 :  mio就是一个模块
# import day5.mio

# 使用模块内的函数,需要通过.语法来进行访问。
# print(day5.mio.add(10, 20))
# print(day5.mio.sub(100, 200))

# import as 语法, 导入文件,并且重命名
# import day5.mio as func  # 导入文件day5.mio 并且重命名为func
# print("func.add(10, 20) : ", func.add(10, 20))
# print("func.sub(100, 200) : ", func.sub(100, 200))
#

# 语法:从模块中,导入某一个函数/类/变量 from xxx import XXX
from day5.mio import add

# 使用add函数
print(add(100, 200))
```

五、异常

5.1 概念

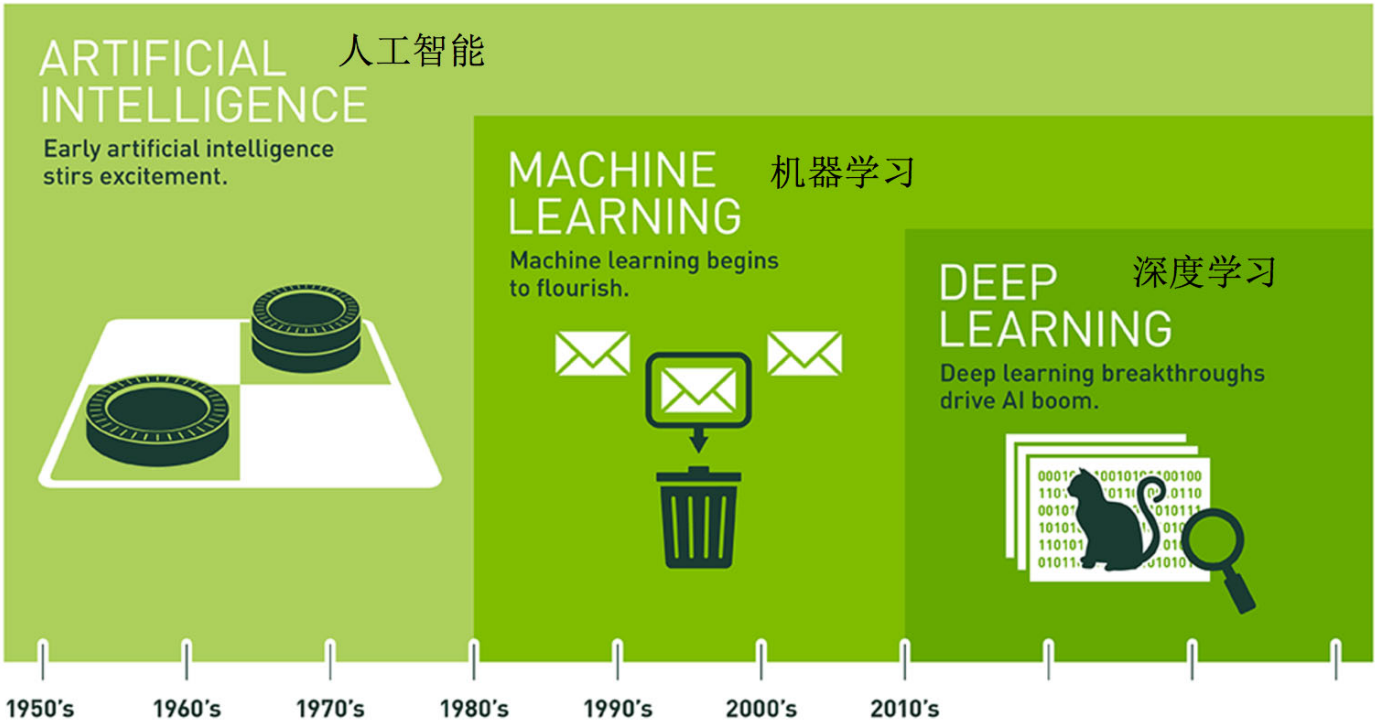
异常就是错误，会导致程序的终止。后面的代码也不会执行了。

5.2 异常处理

```
捕获所有异常:
格式:
try:
    有可能发生异常的代码;
except Exception as e:
    code

含义:
    捕获所有的异常,并且把捕获到的异常,命名为 e

demo:
print("---------------error start ------------------")

try:
    file = open("./test.c", "w")
    print(name) # 没有name 这个变量,这里也是一个异常
except Exception as e:
    print(e)
print("----------------error end --------------------")
```

六、人工智能

6.1 人工智能历史



1950 ~ 1980 ： 简单的机械运动,人工智能。

1980 ~ 2010： 机器学习： 垃圾邮件分类

2010 ~ 至今： 深度学习： 处理图片类型的数据

6.2 机器学习

概念： 机器学习是从数据中自动分析获得规律（模型），并利用规律对未知数据进行预测

6.3 为什么需要机器学习?

解放生产力：

智能客服： 淘宝

辅助医疗： 治疗机器人

6.4 人工智能应用

人工基本上嵌入到生活中的方方面面了，物流，医疗， 教育，生活家居....

预测：

1. 需求量预测：
2. 店铺销量预测

自然语言：

1. 翻译
2. 文本分类
3. 情感分析
4. 声音文字转换

图像：

1. 车牌识别
2. 人脸识别
3. 人脸追踪

七、人工智能数据

7.1 数据来源

i. BOSS给出的数据

ii. 花钱购买数据

iii. 数据网站

1. kaggle：https://www.kaggle.com/datasets
2. uci：http://archive.ics.uci.edu/ml/index.php
3. sklearn: 模块自带简单数据集 --- 方便进行学习

iv. 爬虫工程师派去数据。

7.2 数据类型

5136	5123
10276	10245
15416	15367
20556	20482
25696	25604
30836	30726
35976	35841
41116	40963
46256	46085
51396	51208

特定范围内的汽车数量、人口数量、班级数

数据1

离散型数据：特点：全部由整数组成，通常是记录数据个数的，所以也叫做计数数据。

猫、狗？

0, 1

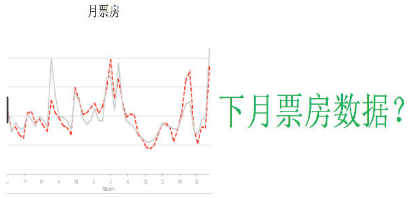
1, 0

0.067732	3.176513
0.427810	3.816464
0.995731	4.550095
0.738336	4.256571
0.981083	4.560815
0.526171	3.929515
0.378887	3.526170
0.033859	3.156393
0.132791	3.110301
0.138306	3.149813
0.247809	3.476346
0.648270	4.119688
0.731209	4.282233

特定范围内的票房数、长度、重量

数据 2

连续性数据：在一定的范围内，可以细分，增加精确度。



7.3 数据集

数据集： 特指值 + 目标值

特征值 目标值

样本	房子面积	房子位置	房子楼层	房子朝向	目标值
数据1	80	9	3	0	80
数据2	100	9	5	1	120
数据3	80	10	3	0	100

目标值： 模型最终想要的结果，就是目标值。

特征值： 会影响到目标值的特真正，就叫做特指值。

样本： 每一行数据，称之为一个样本。

7.4 sklearn自带数据集 ---- 鸢尾花

sklearn模块中的数据： api: sklearn.datasets

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'
['setosa' 'versicolor' 'virginica']
[
 [5.1  3.5  1.4  0.2]
 [4.9  3.   1.4  0.2]
 [4.7  3.2  1.3  0.2]
 [4.6  3.1  1.5  0.2]
 [5.   3.6  1.4  0.2]]
[0
 0
 0
 0
 0]
```

```python
# 鸢尾花数据集
from sklearn.datasets import load_iris
"""
load_iris函数
数据简介:
                花萼长度          花萼宽度          花瓣长度        花瓣宽度
    特征值: sepal length, sepal width, petal length, petal width

                山鸢尾      多色鸢尾      加利福尼亚鸢尾
    目标值: setosa,    versicolor,      virginica

数据返回值:
    return Bunch(data=data,: 特征值
                 target=target, 目标值
                 target_names=target_names, 目标值名称
                 feature_names=feature_names 特征值名称)
"""

```

```python

# 访问数据
def main():

    # 1. 获取鸢尾花的数据
    lr = load_iris() # lr 是一个 Bunch对象.

    # 2. 访问特征值与目标值
    x = lr.data
    y = lr.target

    # 3. 打印结果
    print(lr.feature_names)
    print(x[:5])
    print(lr.target_names)
    print(y[:5])
    return 0


main()
```

7.5 数据集分割

7.5.1 概念

原始数据集需要将数据划分为: 训练集 + 测试集

训练集: 通过数据,训练模型

测试集: 测试训练出来的模型的准确度.

划分比例:

训练集	70%	75%	80%
测试集	30%	25%	20%

7.5.2 API

```python
# 划分数据集的 api
from sklearn.model_selection import train_test_split
"""
train_test_split函数   train: 训练集   test:测试集   split
函数功能:
    分割数据集的训练集 +  测试集

```

```
函数原型:
    def train_test_split(x, y,
                         test_size=None,
                         random_state=None):
函数参数:
    x : 训练集的特征值
    y : 训练集的目标值
    test_size: float类型的数据 测试集的比例
    random_state: 随机数种子

函数返回值:
    x_train: 训练集的特征值
    x_test : 测试集的特征值
    y_train: 训练集的目标值
    y_test : 测试集的目标值
"""


#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    // 计算机中的随机数都是根据一个随机数种子计算出来的.
    // srand(): 设置一个随机数种子

    srand((unsigned int)time(NULL));

    // C语言中: rand()函数, 产生一个随机数
    int number = rand() % 100;

    printf("number = %d\n", number);

    return 0;
}
```

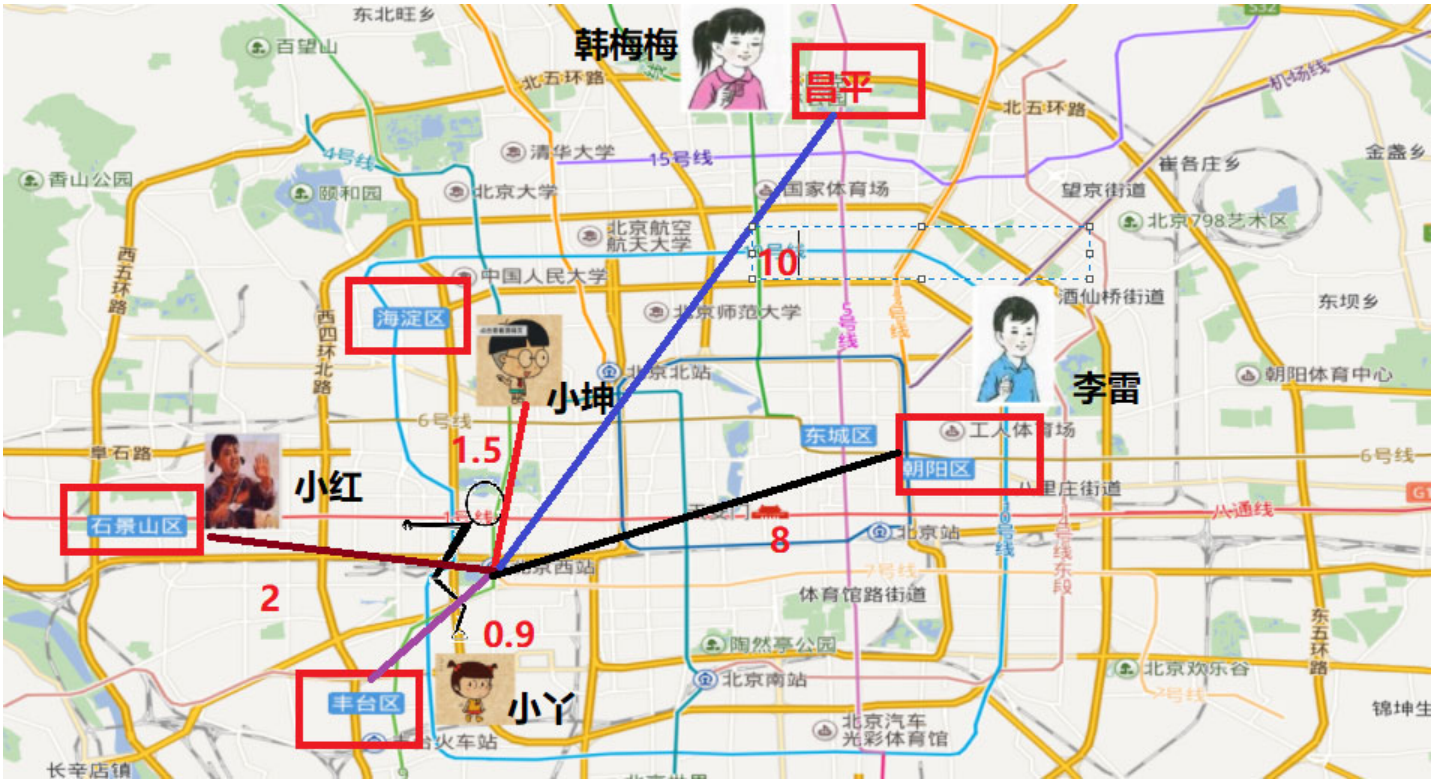
7.5.3 鸢尾花数据集分割

```
# 划分数据集的 api
from sklearn.model_selection import train_test_split

```

```python
# 导入鸢尾花的数据集
from sklearn.datasets import load_iris


def main():

    # 1.获取鸢尾花数据集
    lr = load_iris()

    # 2.确定数据的特征值和目标值
    x = lr.data
    y = lr.target

    # 3.对数据集进行分割处理
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

    # 4.打印结果
    print("x_train : ", x_train)
    print("x_test : ", x_test)
    print("y_train : ", y_train)
    print("y_test : ", y_test)
    return 0


main()
```

八、K-近邻算法

8.1前言



8.2 概念

在一个样本空间中,选取K个最相近的点，如果这K个点中，大部分是属于同一个类别的，那么我认为，我也是属于这个类别。

8.3 距离公式

比如说，a(a1,a2,a3),b(b1,b2,b3)

$$\sqrt{(a1-b1)^2+(a2-b2)^2+(a3-b3)^2}$$

8.4 电影类别的计算

	特征值		目标值	
电影名称	打斗镜头	接吻镜头	电影类型	与未知电影的距离
California Man	3	104	恋爱类型	20.5
He's not Really into dues	2	100	恋爱类型	18.7
Beautiful Woman	1	81	恋爱类型	19.2
Kevin Longblade	101	10	武术类型	115.3
Robo Slayer 3000	99	5	武术类型	117.4
Amped II	98	2	武术类型	118.9
?	18	90		

选取K个最近的点,如果在这K个最相近的点中,大部分属于同一个类别,那么我也认为我是属于这个类别的.

K 值选取特点: 尽量选取奇数个K ,目的:方便类别的甄选

8.5 knn算法API

```
1 from sklearn.neighbors import KNeighborsClassifier
```

```
"""
KNeighborsClassifier类

__init__成员函数

函数原型:
    def __init__(self, n_neighbors=5);

函数参数:
    n_neighbors:  K值,表示选取多少个最相近的邻居

fit成员函数

函数功能:
    拟合数据(训练集的特征值, 训练集的目标值),制作模型.

函数原型:
    fit(self, X, y);

函数参数:
    X: 训练集的特征值
    y: 训练集的目标值


predict成员函数

函数功能:
    根据给出的特征值,对数据进行预测,得到目标值

函数原型:
    predict(self, X);

函数参数:
    X:  需要预测的数据的特征值

返回值:
    列表: 每一个样本的预测结果

score成员函数

函数功能:
    测试模型的准确度.

函数原型:
```

```
    score(self, X, y, sample_weight=None);

函数参数:
    X: 测试集数据的特征值
    y: 测试集数据的目标值

返回值:
    模型准确度.

"""
```

8.6 鸢尾花样本预测

```
# target 鸢尾花:[3.6, 2.7, 1.5, 0.8] 预测一下这个鸢尾花的种类
"""
<1>. 通过鸢尾花数据集,制作一个模型
<2>. 使用predict函数对鸢尾花种类进行预测
"""
# 导入鸢尾花的数据集
from sklearn.datasets import load_iris
# 导入划分数据集的方法
from sklearn.model_selection import train_test_split
# 导入KNN算法模型的类
from sklearn.neighbors import KNeighborsClassifier


def main():

    # 1.获取鸢尾花数据集
    lr = load_iris()

    # 2.确定数据的特征值与目标值
    x = lr.data
    y = lr.target

    # 3.分割数据集,将数据集划分为训练集和测试集
    x_train,x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

    # 4.实例化一个算法对象
    estimate = KNeighborsClassifier(n_neighbors=7)

```

```
    # 5.拟合数据,制作模型
    estimate.fit(x_train, y_train)

    # 6.计算模型的准确度
    score = estimate.score(x_test, y_test)

    print("score : ", score)

    # 7.对数据进行预测
    data = [[3.6, 2.7, 1.5, 0.8],[3.6, 2.7, 1.5, 0.8],[3.6, 2.7, 1.5, 0.8],[3.6, 2.7, 1.

    y_predict = estimate.predict(data)
    number = 1
    for i in y_predict:
        print("第 %d 样本: %s" %(number, lr.target_names[i]))
        number += 1

    print(y_predict)
    return 0


main()
```