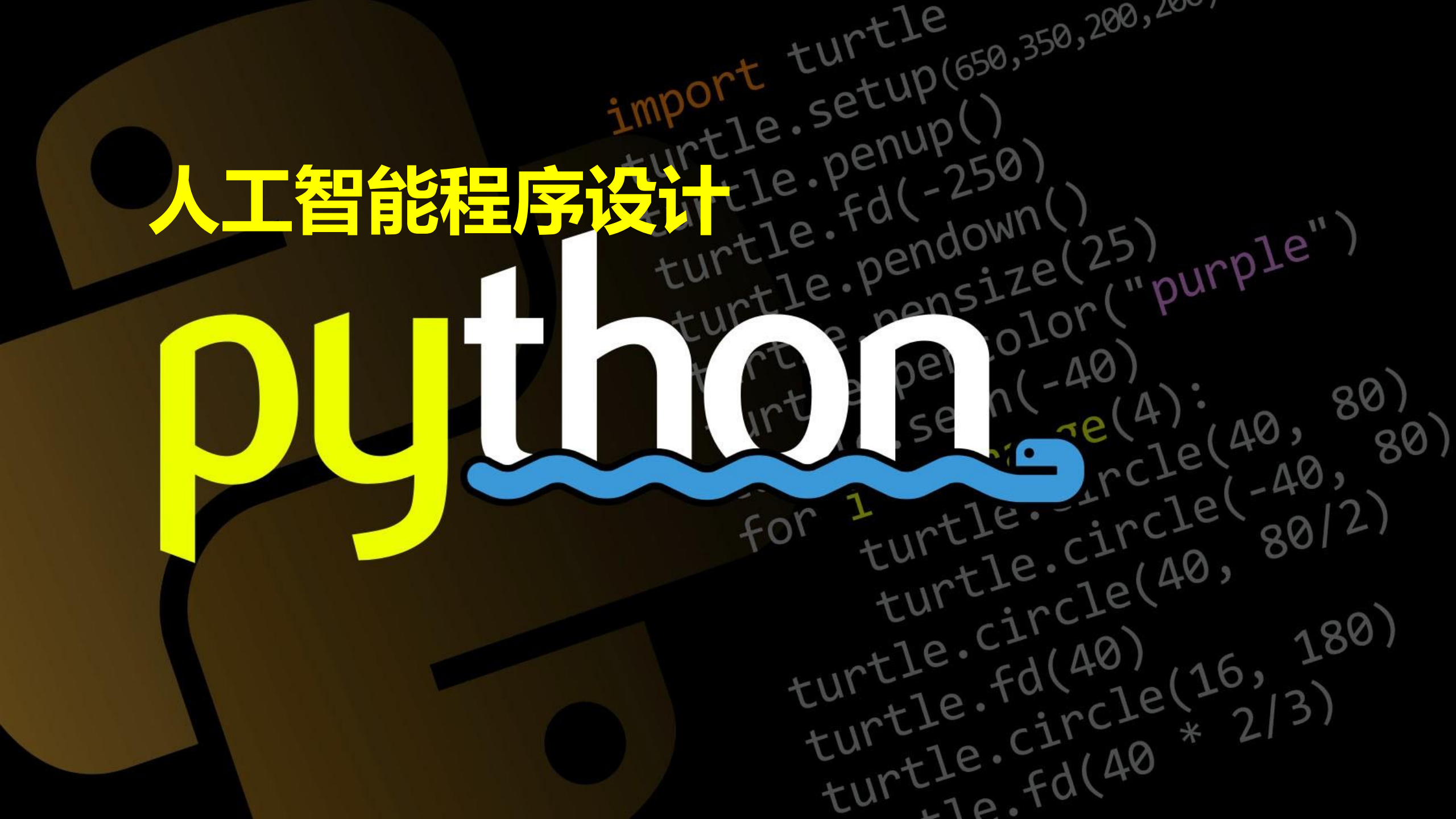
人工智能程序设计


python
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
turtle.seth(-40)
for 1 in range(4):
turtle.circle(40, 80)
turtle.circle(-40, 80)
turtle.fd(40)
turtle.circle(40, 80/2)
turtle.fd(40)
turtle.circle(16, 180)
turtle.fd(40 * 2/3)

人工智能程序设计

3.2 管理列表

北京石油化工学院 人工智能研究院

刘 强

3.2 管理列表

在实际编程中，我们经常需要对列表进行动态管理，包括添加新元素、删除不需要的元素、修改现有元素等

```
fruits = ["苹果", "香蕉"]
fruits.append("橙子")
```

3.2.1 添加元素

在列表末尾添加元素

使用append()方法在列表末尾添加单个元素：

```
fruits = ["苹果", "香蕉"]
fruits.append("橙子")
print(fruits)  # 输出：['苹果', '香蕉', '橙子']

## 添加不同类型的元素
numbers = [1, 2, 3]
numbers.append("四")
print(numbers)  # 输出：[1, 2, 3, '四']
```

3.2.1 添加元素

在指定位置插入元素

使用insert()方法在指定位置插入元素：

```
colors = ["红", "蓝"]
colors.insert(1, "绿")  # 在索引1的位置插入"绿"
print(colors)  # 输出：['红', '绿', '蓝']

## 在开头插入
colors.insert(0, "黄")
print(colors)  # 输出：['黄', '红', '绿', '蓝']
```

3.2.1 添加元素

扩展列表

使用extend()方法将另一个列表的所有元素添加到当前列表：

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
list1.extend(list2)
print(list1)  # 输出：[1, 2, 3, 4, 5, 6]

## 扩展字符串（字符串也是可迭代对象）
letters = ["a", "b"]
letters.extend("cd")
print(letters)  # 输出：['a', 'b', 'c', 'd']
```

3.2.2 删除元素

根据值删除元素

使用remove()方法删除第一个匹配的元素：

```
animals = ["猫", "狗", "鸟", "猫"]
animals.remove("猫")  # 只删除第一个"猫"
print(animals)  # 输出：['狗', '鸟', '猫']
```

3.2.2 删除元素

根据索引删除元素

使用pop()方法删除指定位置的元素，并返回该元素：

```
numbers = [10, 20, 30, 40]
removed = numbers.pop(1)  # 删除索引1的元素
print(f"删除的元素：{removed}")  # 输出：删除的元素：20
print(numbers)  # 输出：[10, 30, 40]

## 删除最后一个元素
last = numbers.pop()
print(f"删除的最后元素：{last}")  # 输出：删除的最后元素：40
```

3.2.2 删除元素

使用del语句删除

```
subjects = ["数学", "语文", "英语", "物理"]
del subjects[1]  # 删除索引1的元素
print(subjects)  # 输出：['数学', '英语', '物理']

## 删除切片
del subjects[1:3]
print(subjects)  # 输出：['数学']
```

3.2.2 删除元素

清空列表

使用clear()方法可以清空列表中的所有元素：

```
temp_list = [1, 2, 3, 4, 5]
temp_list.clear()
print(temp_list)  # 输出：[]
```

3.2.3 修改元素

修改单个元素

直接通过索引赋值来修改元素：

```
grades = [85, 92, 78, 96]
grades[2] = 88  # 修改索引2的元素
print(grades)  # 输出：[85, 92, 88, 96]
```

3.2.3 修改元素

修改多个元素

通过切片修改多个元素：

```
letters = ["a", "b", "c", "d", "e"]
letters[1:4] = ["B", "C", "D"]
print(letters)  # 输出：['a', 'B', 'C', 'D', 'e']

## 替换为不同数量的元素
letters[1:4] = ["X", "Y"]
print(letters)  # 输出：['a', 'X', 'Y', 'e']
```

3.2.4 列表排序与顺序反转

永久排序

使用**sort()方法**对列表进行永久排序：

```
numbers = [3, 1, 4, 1, 5, 9, 2, 6]
numbers.sort()
print(numbers)  # 输出：[1, 1, 2, 3, 4, 5, 6, 9]
## 逆序排序
numbers.sort(reverse=True)
print(numbers)  # 输出：[9, 6, 5, 4, 3, 2, 1, 1]
## 字符串排序
names = ["张三", "李四", "王五", "赵六"]
names.sort()
print(names)  # 输出：['张三', '李四', '王五', '赵六']
```

3.2.4 列表排序与顺序反转

临时排序

使用**sorted()函数**返回排序后的新列表，原列表不变：

```
original = [3, 1, 4, 1, 5]
sorted_list = sorted(original)
print(f"原列表：{original}")      # 输出：原列表：[3, 1, 4, 1, 5]
print(f"排序后：{sorted_list}")   # 输出：排序后：[1, 1, 3, 4, 5]
```

3.2.4 列表排序与顺序反转

列表反转

使用**reverse()方法**可以反转列表中元素的顺序：

```
numbers = [1, 2, 3, 4, 5]
numbers.reverse()
print(numbers)  # 输出：[5, 4, 3, 2, 1]
```

实践练习

练习 3.2.1：图书管理系统

创建一个图书列表，实现添加图书、删除指定图书、修改图书信息的功能。

练习 3.2.2：购物车操作

模拟购物车操作：添加商品、删除商品、修改商品数量，并计算总价。

练习 3.2.3：排行榜管理

创建一个游戏分数列表，实现排序、查找最高分和最低分、计算平均分的功能。