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

基本程序框架

北京石油化工学院 人工智能研究院

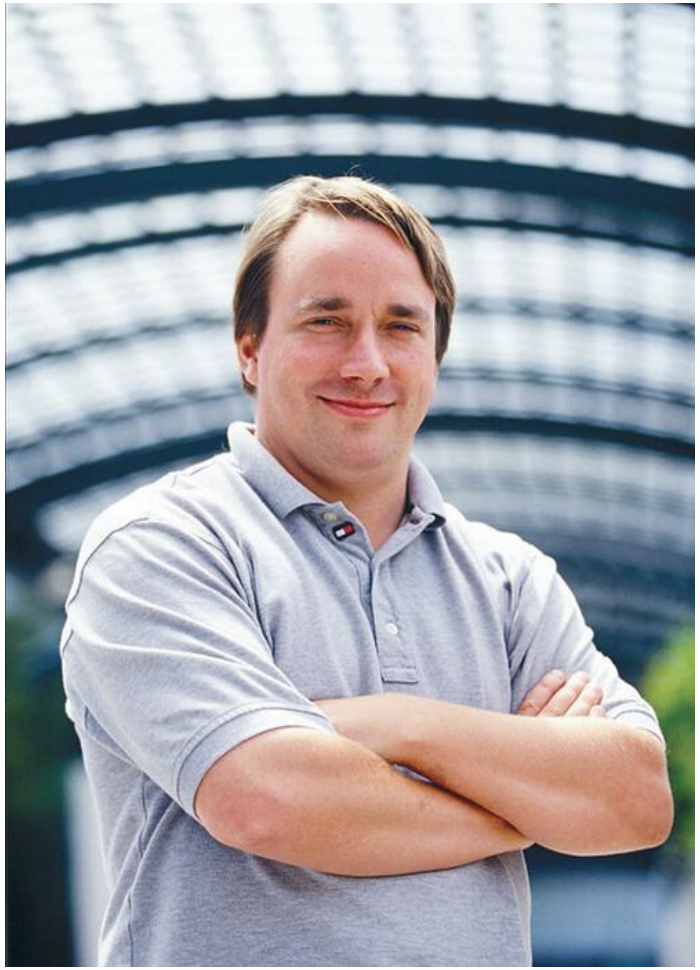
刘 强

代码胜于雄辩。
Talking is cheap. Show me the code.

林纳斯·托瓦兹

Linus Torvalds

Linux操作系统之父



示例：圆面积计算

```
# 导入数学模块
import math
# 定义计算圆面积的函数
def calculate_circle_area(radius):
    """
    计算圆的面积
    参数: radius - 圆的半径
    返回: 圆的面积
    """
    area = math.pi * radius ** 2
    return area
```

模块导入

函数定义

```
# 主函数
def main():
    # 获取用户输入
    radius = float(input("请输入圆的半径: "))
    # 检查输入是否有效
    if radius <= 0:
        print("半径必须大于0！")
    else:
        # 计算并输出结果
        area = calculate_circle_area(radius)
        print(f"半径为 {radius} 的圆的面积是: {area:.2f}")
# 程序入口点
if __name__ == "__main__":
    main()
```

主函数

程序入口

Python程序的典型结构

1. 模块导入：`import math` 导入数学模块
2. 函数定义：定义具体功能的函数
3. 主函数：组织程序的主要逻辑
4. 程序入口：`if __name__ == "__main__":` 确保脚本直接运行时执行

Python程序的执行流程

1. Python解释器从上到下逐行执行
2. 首先导入math模块
3. 定义函数（但不执行）
4. 遇到 `if __name__ == "__main__"`时执行main()函数
5. main()函数调用其他函数完成任务

Python程序的语法元素

1. 变量和数据类型

```
radius = 3.5    # 浮点数变量
area = 38.48    # 浮点数变量
```

2. 运算符

```
area = math.pi * radius ** 2  # 乘法(*)和幂运算(**)
radius <= 0                   # 比较运算符
```

3. 函数

```
def calculate_circle_area(radius):  # 函数定义
    return area                     # 返回值
area = calculate_circle_area(5)     # 函数调用
```

Python程序的语法元素

4. 控制结构

```
if radius <= 0:        # 条件判断
    print("错误信息")
else:
    print("正确结果")
```

5. 模块和导入

```
import math          # 导入模块
math.pi              # 使用模块中的常量
```

6. 注释

```
# 这是单行注释
"""
这是多行注释
用于函数说明
"""
```

程序的输入输出操作

input()函数 - 获取用户输入

```
# 基本用法
name = input("请输入您的姓名: ")


# 类型转换
radius = float(input("请输入半径: ")) # 转换为浮点数
age = int(input("请输入年龄: "))      # 转换为整数
```

重要提示：`input()`函数返回的始终是字符串，需要数字时必须进行类型转换。

程序的输入输出操作

print()函数 - 输出信息

```
# 基本输出
print("Hello, Python!")        # 打印字符串 ” Hello, Python!”

# 格式化输出
print(f"圆的面积是: {area:.2f}") # 输出浮点数，保留小数点后2位数值

# 输出多个值
print("姓名:", name, "年龄:", age) # 输出姓名和年龄
```

示例：平方根的计算

导入模块
定义函数

主函数

程序入口点

```python
import math

def greet(name):
    print(f"Hello, {name}!")

def main():
    # 获取用户输入
    username = input("Please enter your name: ")
    # 调用函数
    greet(username)

    # 计算示例
    number = float(input("Enter a number to calculate its square root: "))
    square_root = math.sqrt(number)
    print(f"The square root of {number} is {square_root}")

main()
```

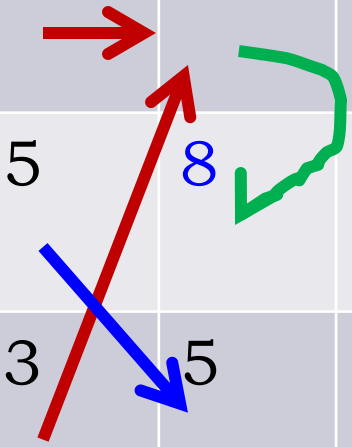
示例：裴波那契数列的计算

斐波那切兔子问题 (Fibonacci rabbit problem)—**一道著名数列难题**：

意大利数学家裴波那契(Fibonacci , L.)在他的名著《算法之书》中提出的一个问题：每对大兔每月能生产一对小兔，而每对小兔生长两个月就成为大兔。由一对兔子开始，一年后可以繁殖成多少对兔子?

示例：裴波那契数列的计算

月份	1	2	3	4	5	6	7	8	9	10	11	12
大兔	0	0	1	1	2	3	5	8	大兔=上月大兔+2月兔			
1月兔	1	0	1	1	2	3	5	8	1月兔=当月大兔			
2月兔	0	1	0	1	1	2	3	5	2月兔=上月1月兔			
数量总计	1	1	2	3	5	8	13	21				

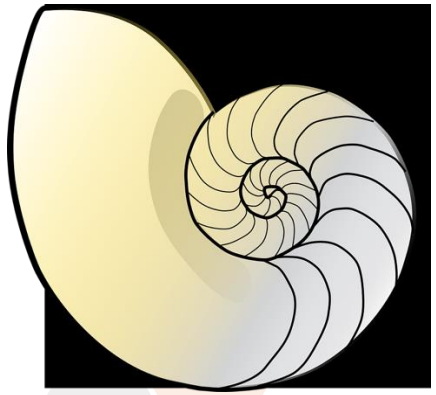


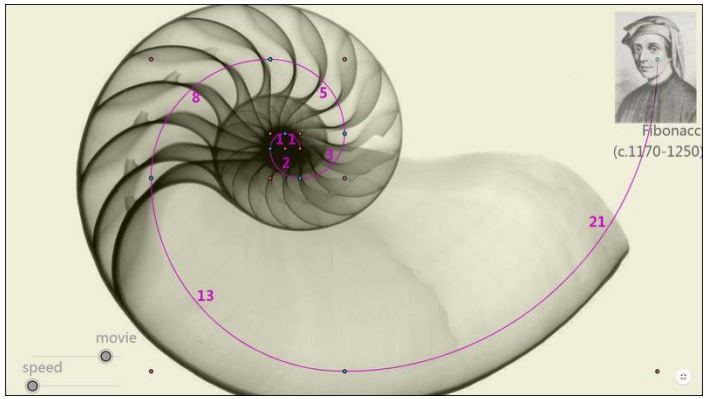
示例：裴波那契数列的计算

裴波那契数列是一个经典的数列，其中每个数字是前两个数字的和，通常定义为Fib(0) = 0, Fib(1) = 1,

并且对于n > 1，

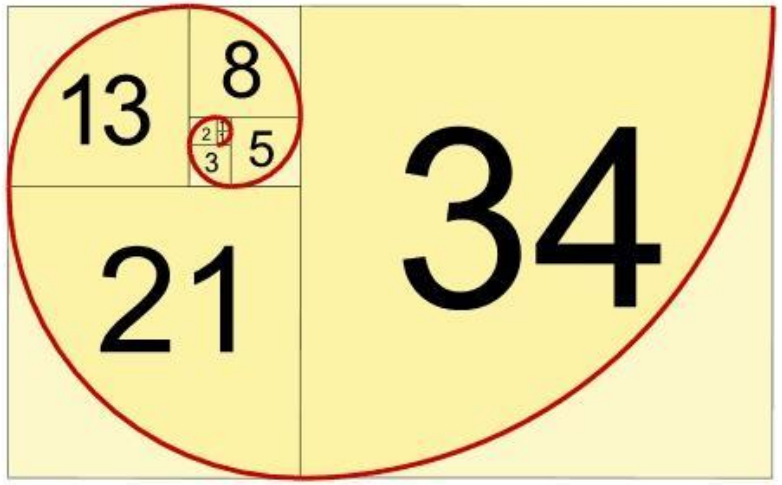
有Fib(n) = Fib(n-1) + Fib(n-2)。





做个螺旋

用这些数作为边长来画正方形，就得到漂亮的螺旋图象：



留意正方形整齐地形成螺旋形。

例如，5 和 8 是 13、8 和 13 是 21 等等。

示例：裴波那契数列的计算

定义函数

```python
# 定义一个函数来计算斐波那契数列的第n项
def fibonacci(n):
    # 定义前两个斐波那契数
    a, b = 0, 1
    # 使用for循环来计算斐波那契数列的第n项
    for i in range(2, n + 1):
        # 赋值语句来更新斐波那契数
        a, b = b, a + b
    # 返回斐波那契数列的第n项
    return a
```

Fib(0) = 0, Fib(1) = 1
从第0月开始计数，第1月有1对小兔

示例：裴波那契数列的计算

主函数

```python
# 主函数
def main():
    # 变量n用于存储用户希望计算的斐波那契数列项
    n = int(input("请输入要计算的斐波那契数列项(正整数): "))

    # 检查用户输入是否为正整数
    if n < 0:
        print("输入错误，请输入一个正整数。")
    else:
        # 调用fibonacci函数并打印结果
        result = fibonacci(n)
        print(f"斐波那契数列的第{n}项是: {result}")

# 程序入口点
if __name__ == "__main__":
    main()
```

程序入口点

程序框架解析：裴波那契数列的计算

```python
# 定义一个函数来计算斐波那契数列的第n项
def fibonacci(n):
    # 定义前两个斐波那契数
    a, b = 0, 1
    # 使用for循环来计算斐波那契数列的第n项
    for i in range(2, n + 1):
        # 赋值语句来更新斐波那契数
        a, b = b, a + b
    # 返回斐波那契数列的第n项
    return a
```

1.函数定义：使用 def 关键字定义了一个名为 fibonacci 的函数，它接受一个参数 n。
2.变量和数据类型：在函数内部，定义了两个变量 a 和 b，初始化为0和1，表示斐波那契数列的前两项。
3.循环控制语句：使用 for 循环来迭代计算斐波那契数列的后续项，range(2, n + 1) 产生从2到n的序列。
4.赋值语句：在循环体内部，使用赋值语句 a, b = b, a + b 来更新 a 和 b 的值，这是Python的多重赋值特性。
5.返回值：使用 return 语句返回斐波那契数列的第 n 项。

程序框架解析：裴波那契数列的计算

```python
# 主函数
def main():
    # 变量n用于存储用户希望计算的斐波那契数列项
    n = int(input("请输入要计算的斐波那契数列项(正整数): "))

    # 检查用户输入是否为正整数
    if n < 0:
        print("输入错误，请输入一个正整数。")
    else:
        # 调用fibonacci函数并打印结果
        result = fibonacci(n)
        print(f"斐波那契数列的第{n}项是: {result}")

# 程序入口点
if __name__ == "__main__":
    main()
```

1.主函数：main 函数封装了程序的主要逻辑，包括用户输入处理和调用 fibonacci 函数。
2.条件语句：使用 if 语句检查用户输入是否为正整数。
3.输入和输出：使用 input 函数获取用户输入，并使用 print 函数输出结果。
4.程序入口点：if __name__ == "__main__": 确保当脚本被直接运行时，main 函数会被调用。

动手试一试：斐波那契数列的计算

```python
# 定义一个函数来计算斐波那契数列的第n项
def fibonacci(n):
    # 定义前两个斐波那契数
    a, b = 0, 1
    # 使用for循环来计算斐波那契数列的第n项
    for i in range(2, n + 1):
        # 赋值语句来更新斐波那契数
        a, b = b, a + b
    # 返回斐波那契数列的第n项
    return a
```

```python
# 主函数
def main():
    # 变量n用于存储用户希望计算的斐波那契数列项
    n = int(input("请输入要计算的斐波那契数列项(正整数): "))

    # 检查用户输入是否为正整数
    if n < 0:
        print("输入错误，请输入一个正整数。")
    else:
        # 调用fibonacci函数并打印结果
        result = fibonacci(n)
        print(f"斐波那契数列的第{n}项是: {result}")

# 程序入口点
if __name__ == "__main__":
    main()
```

实践练习

练习 1.4.1：圆周长计算

将圆面积计算程序修改为计算圆的周长。

练习 1.4.2：三角形面积计算

编写一个计算三角形面积的程序，要求用户输入底边和高，然后输出三角形的面积。