2023hw3

2023 年 10 月 30 日

1. 鸡兔同笼

[4]:
```
a, b = input().split()

def solve(a, b):
    if a < 0 or b < 0:
        print("Data Error!")
        return
    rabbits = b/2 - a
    chi = a - rabbits
    if chi - int(chi) != 0 or chi < 0 or rabbits < 0:
        print("Data Error!")
        return
    print(f"有{int(chi)}只鸡, {int(rabbits)}只兔")

    return

solve(int(a),int(b))
```

```
-24 12
Data Error!
```

2. 中国古代数学问题——物不知数

[ ]:
```
n = eval(input())
flag = 0
for i in range(n+1):
    if (i%3 == 2) and (i%5 == 3) and (i%7 == 2):
        print(i)
        flag = 1
if not flag:
    print('No solution!')
```

3. 中国古代数学问题——二鼠打洞

[ ]:
```
wall = eval(input())
mouse1 = 1
mouse2 = 1
day = 0
sum1 = 0
sum2 = 0
while wall > 0:
    if wall - mouse1 - mouse2 <= 0:
        t = wall / (mouse1 + mouse2)
        sum1 += (t * mouse1)
        sum2 += (t * mouse2)
        day += 1
        break
    else:
        wall -= (mouse1 + mouse2)
        day += 1
        sum1 += mouse1
        sum2 += mouse2
        mouse1 *= 2
        mouse2 *= 0.5
print(day)
print(round(sum2, 1), round(sum1, 1), sep=' ')
```

4. 中国古代数学问题——李白买酒

[ ]:
```
wine = 0
for i in range(5):
    wine += 1
    wine /= 2
print(wine)
```

5. 中国古代数学问题——宝塔上的琉璃灯

[6]:
```
floor = 765 / (2**8 - 1)
for i in range(8):
    print(int(floor*(2**i)))
```

```
3.0
6.0
12.0
```

```
24.0
48.0
96.0
192.0
384.0
```

6. 计算圆周率——割圆法

[5]:
```python
# --------      -------    --------
# @Author : 赵广辉
# @Contact: vasp@qq.com
# @Company: 武汉理工大学
# @Version: 1.0
# @Modify : 2022/06/13 11:33
# Python 程序设计基础，赵广辉，高等教育出版社，2021
# Python 程序设计基础实践教程，赵广辉，高等教育出版社，2021
# --------      -------    --------

# = 周长/(2* 圆的半径) 得到  的近似值。
# 半径为 1 的圆内接正 6 边形边长也是 1
# 边长  side_length
# 半径  radius
# 圆周率 pi
# 三角形的高 height
import math


def cutting_circle(times):    # times 为分割次数
    side_length = 1           # 初始边长
    edges = 6                 # 初始边数
    def length(x):
        h = math.sqrt(1 - (x / 2)**2)
        result = math.sqrt((x / 2)**2 + (1-h)**2)
        return result

    for i in range(times):
        side_length = length(side_length)
        edges = edges * 2
        pi = side_length*edges/2
```

```
    return edges, pi


if __name__ == '__main__':
    times = int(input())           # 割圆次数
    print('分割{}次，边数为{}，圆周率为{:.6f}'.format(times,␣
 ↪*cutting_circle(times)))          # 圆周率
    print('math 库中的圆周率常量值为{:.6f}'.format(math.pi))
```

```
4
分割 4 次，边数为 96，圆周率为 3.141032
math 库中的圆周率常量值为 3.141593
```

7. 计算圆周率——无穷级数法

[6]:
```
def leibniz_of_pi(error):
    num = 0
    i = 1
    while 1/(2*i-1) >= error:
        num += 4*(-1)**(i+1)/(2*i-1)
        i += 1
    return num

if __name__ == '__main__':
    threshold = float(input())
    print("{:.8f}".format(leibniz_of_pi(threshold)))
```

```
2e-7
3.14159225
```

8. 计算圆周率——蒙特卡洛法

[ ]:
```
import random
sd = eval(input())
num = eval(input())

random.seed(sd)

n = 0
for i in range(num):
    x, y = random.uniform(-1,1), random.uniform(-1,1)
```

```
    if (x**2 + y**2) <= 1:
        n += 1

m_pi = n / num*4
print(m_pi)
```

9. 素数问题

[24]:
```
# --------      -------    --------
# @Author : 赵广辉
# @Contact: vasp@qq.com
# @Company: 武汉理工大学
# @Version: 1.0
# @Modify : 2022/06/13 11:33
# Python 程序设计基础，赵广辉，高等教育出版社，2021
# Python 程序设计基础实践教程，赵广辉，高等教育出版社，2021
# --------      -------    --------

def question_judge(question):
    """ 接收一个字符串为参数，根据参数值判断问题类型，调用合适的函数进行操作。"""
    if question == '素数':        # 如果输入”素数“，再输入一个正整数 n，输出不大于 n
的所有素数
        n = int(input())
        output_prime(n)          # 输出素数
    elif question == '回文素数':
        n = int(input())
        palindromic_prime(n)    # 输出回文素数
    elif question == '反素数':
        n = int(input())
        reverse_prime(n)  # 输出反素数
    elif question == '哥德巴赫猜想':
        n = int(input())
        goldbach_conjecture(n)
    else:
        print('输入错误')



def is_prime(n):
```

```
    """ 判断素数的函数，接收一个正整数为参数，参数是素数时返回 True，否则返回 False
    减小判定区间，减少循环次数，提升效率"""
    flag = True
    if n == 1:
        flag = False
    else:
        for i in range(2, int(n**0.5) + 1):
            if n % i == 0:
                flag = False
                break
    return flag



def output_prime(number):
    """ 接收一个正整数为参数，遍历从 0 到 number 之间的所有整数
    在一行中输出不大于 number 的所有素数，函数无返回值"""
    for i in range(1, number+1):
        if is_prime(i):
            print(i, end=' ')



def palindromic(num):
    """ 接收一个数字为参数，判定其是否为回文数，返回布尔值。"""
    return str(num) == str(num)[::-1]


def palindromic_prime(number):
    """ 接收一个正整数参数 number，遍历从 0 到 number 之间的所有整数，
    若某个数是素数，且转为字符串后是回文字符串，则称其为回文素数
    找出并在同一行中从小到大输出小于 number 的所有回文素数，各数字间用一个空格分隔，
    函数无返回值"""
    for i in range(1, number):
        if palindromic(i) and is_prime(i):
            print(i, end=' ')
```

```
def reverse_num(num):
    """ 接收一个整数，返回其逆序字符串对应的整数"""
    return int(str(num)[::-1])


def reverse_prime(number):
    """ 接收一个正整数参数，找出并在同一行内输出所有小于 number 的反素数，数字间用一个空格分隔。
    反素数指某数 i 及其逆序数都是素数，但数 i 对应的字符串不是回文字符串
    函数无返回值"""
    for i in range(1, number):
        if not palindromic(i) and is_prime(i) and is_prime(reverse_num(i)):
            print(i, end=' ')



def goldbach_conjecture(num):
    """ 哥德巴赫猜想，接收一个不小于 4 的正整数为参数。
    当参数为不小于 4 的偶数时，将其分解为两个素数的加和，按小数 + 数的格式输出。
    有多种组合时全部输出，但不输出重复的组合，例如输出 8=3+5，不输出 8=5+3。
    参数为奇数或小于 4 时，输出'Data error!'
    """
    if num < 4 or num%2 != 0:
        print('Data error!')
    else:
        for i in range(2, num // 2 + 1):
            j = num - i
            if is_prime(i) and is_prime(j):
                print(f'{num}={i}+{j}')



if __name__ == '__main__':
    problems = input()
    question_judge(problems)
```

哥德巴赫猜想

30

30=7+23

30=11+19

30=13+17

10. 计算圆周率

[15]:
```python
# --------      -------    --------
# @File   : 计算圆周率.py
# @Author : 赵广辉
# @Contact: vasp@qq.com
# @Company: 武汉理工大学
# @Version: 1.0
# @Modify : 2021/10/30 23:18
# Python 程序设计基础，高等教育出版社
# --------      -------    --------
import math
import random
import matplotlib.pyplot as plt


def type_judge(pi_type):
    """ 接收一个字符串为参数，根据参数调用相应函数计算圆周率。    """
    if pi_type == '割圆法':
        times = int(input())  # 输入一个表示边数量的正整数
        return cutting_circle(times)  # 调用函数计算圆周率
    elif pi_type == '无穷级数法':
        threshold = float(input())   # 输入转为浮点数
        return leibniz_of_pi(threshold)  # 调用函数计算圆周率
    elif pi_type == '蒙特卡洛法':
        num = int(input())           # 输入转为整数
        s = int(input())             # 输入随机数种子
        return monte_carlo_pi(num, s)  # 调用函数计算圆周率
    elif pi_type == '梅钦法':
        return machin_of_pi()  # 调用函数计算圆周率
    elif pi_type == '拉马努金法':
        num = int(input())  # 输入转为整数
        return ramanujan_of_pi(num)
    else:
        return f'未找到{pi_type}计算方法'
```

```
def cutting_circle(times):
    """ 参数 times 为分割次数
      = 周长/(2* 圆的半径) 得到   的近似值。
    # 半径为 1 的圆内接正 6 边形边长也是 1
    # 边长  side_length
    # 半径  radius
    # 圆周率 pi
    # 三角形的高 height
    """
    side_length = 1           # 初始边长
    edges = 6                 # 初始边数
    def length(x):
        h = math.sqrt(1 - (x / 2)**2)
        result = math.sqrt((x / 2)**2 + (1-h)**2)
        return result

    for i in range(times):
        side_length = length(side_length)
        edges = edges * 2
        pi = side_length*edges/2

    return pi


def leibniz_of_pi(error):
    """ 接收用户输入的浮点数阈值为参数，用格雷戈里-莱布尼茨级数计算圆周率，返回圆周率
值"""
    pi = 0
    i = 1
    while 1/(2*i-1) >= error:
        pi += 4*(-1)**(i+1)/(2*i-1)
        i += 1
    return pi


def monte_carlo_pi(num, s):
    """ 接收一个表示随机次数的整数和一个整数做随机数种子，用蒙特卡洛法计算圆周率，返回
一个浮点数"""
```

```
    random.seed(s)
    n = 0
    for i in range(num):
        x, y = random.uniform(-1,1), random.uniform(-1,1)
        if (x**2 + y**2) <= 1:
            n += 1

    pi = n / num*4
    return pi

def machin_of_pi():
    """ 用梅钦级数计算圆周率，返回圆周率值"""
    pi = 4*(4*math.atan(1/5) - math.atan(1/239))
    return pi

def ramanujan_of_pi(n):
    """ 接收一个正整数 n 为参数，用拉马努金公式的前 n 项计算圆周率并返回。"""
    s = 0
    for i in range(n+1):
        s += (math.factorial(4*i) * (1103+26390*i)) / ((math.factorial(i))**4 *␣
↪396 ** (4*i))
    pi = 1 / ((2*math.sqrt(2)) / 9801 * s)
    return pi


# 取消以下代码行前面的注释可以调用其中的函数对割圆法和蒙特卡洛法进行演示

# def draw_circle(r, side_num):
#     """ 创建图形和轴，用折线图绘制正多边形。
#     @ 参数 r: 圆的半径
#     @ 参数 side_num: 正多边形的边数
#     """
#     fig, ax = plt.subplots()            # 创建图形和轴
#     plt.subplots_adjust(left=0.1, bottom=0.25)
#     x, y = xy_of_polygon(r, side_num)
#     plt.plot(x, y, lw=2, color='red')  # 设置线宽和颜色
#     ax.set_aspect('equal')              # 设置坐标轴纵横比相等
#     ax.set_xlim(-r-5, r+5)              # 设置 x 轴刻度起止值
#     ax.set_ylim(-r-5, r+5)
```

```
#     plt.draw()                              # 重新绘制多边形
#     plt.show()                              # 显示图形
#
#
# def cutting_circle(times):
#     """
#     接收表示分割次数的整数 n 为参数，计算分割 n 次时正多边形的边数和圆周率值，返回边数和圆周率值。
#     @参数 times: 分割次数
#       = 周长/(2* 圆的半径) 得到   的近似值。
#     # 半径为 1 的圆内接正 6 边形边长也是 1
#     # 边长  side_length
#     # 半径  radius
#     # 圆周率 pi
#     # 三角形的高 height
#     >>> cutting_circle(4)
#     3.14103195089051
#     """
#     side_length = 1           # 初始边长
#     edges = 6                 # 初始边数
#     for i in range(times):
#         height = 1 - math.sqrt(1 - (side_length / 2) ** 2)
#         side_length = math.sqrt(height ** 2 + (side_length / 2) ** 2)
#         edges = edges * 2           # 每割一次，边数量加倍
#         draw_circle(300, edges)  # 调用此函数可演示割圆效果，每次循环绘制一个近似圆，关闭后看下一个
#     pi = side_length * edges / 2
#     return pi
#
#
# def monte_carlo_show(in_circle_lst, out_circle_lst):
#     """
#     @参数 in_circle_lst: 落在圆内的点的坐标列表
#     @参数 out_circle_lst: 落在圆外的点的坐标列表
#
#     """
#     plt.rcParams['font.sans-serif'] = ['SimHei']   # 用来正常显示中文标签
#     plt.rcParams['axes.unicode_minus'] = False     # 用来正常显示负号
#     plt.figure(figsize=(11, 11))                   # 设置画布长宽
```

```
#     x_in_circle = [x[0] for x in in_circle_lst]    # 落在圆内的点的列表 x
#     y_in_circle = [x[1] for x in in_circle_lst]    # 落在圆内的点的列表 y
#     x_out_circle = [x[0] for x in out_circle_lst]  # 落在圆外的点的列表 x
#     y_out_circle = [x[1] for x in out_circle_lst]  # 落在圆外的点的列表 y
#     plt.scatter(x_out_circle, y_out_circle, s=10, facecolors='blue')  # 绘制散点，落在圆外在点颜色用蓝色
#     plt.scatter(x_in_circle, y_in_circle, s=5, facecolors='red')    # 绘制散点，落在圆内在点颜色用红色
#     plt.title('蒙特卡洛法计算圆周率演示')              # 图的标题
#     plt.show()                                      # 显示绘制结果
#
#
# # 给出 turtle 模板，学生填 monte_carlo 代码，帮助学生理解算法，了解 turtle 绘图
# def monte_carlo_pi_turtle(n, s):
#     """ 用 turtle 绘图模拟蒙特卡洛 n 次结果"""
#     random.seed(s)
#     turtle.tracer(1000)
#     turtle.pensize(4)
#     turtle.penup()
#     turtle.goto(-300, -300)
#     turtle.pendown()
#     for i in range(4):
#         turtle.forward(600)
#         turtle.left(90)
#     turtle.forward(300)
#     turtle.pencolor('green')
#     turtle.circle(300)
#     hits = 0  # 落在圆内的计数器初值设为 0
#     for i in range(1, n + 1):
#         x, y = random.uniform(-1, 1), random.uniform(-1, 1)  # 生成两个随机数模拟一个点的坐标
#         pos = (x ** 2 + y ** 2) ** 0.5                       # 计算坐标（x,y）到原点的距离
#         if pos <= 1.0:                                       # 如果距离小于等于 1，点在圆内
#             hits = hits + 1
#             turtle.pencolor('red')                           # 落在圆内在点颜色用红色
```

```
#         else:                                                  # 如果距离大于 1，点在圆外
#             turtle.pencolor('blue')                            # 落在圆内在点颜色用蓝色
#         turtle.penup()
#         turtle.goto(x * 300, y * 300)                          # 画笔抬起并移动到数值放大 400 倍的 x,y 处
#         turtle.pendown()
#         turtle.dot(3)                                          # 画一个半径为 3 的圆点
#         if i % 10000 == 0:                                     # 实验为 10000 的倍数次时
#             turtle.pencolor('black')
#             pi = 4 * (hits / i)                                # 根据落在圆内外的点数量估算 PI 值
#             turtle.penup()                                     # 画笔抬起
#             turtle.goto(320, 150 - i // 1000 * 30)             # 移动到区域外记录当前 PI 值
#             turtle.pendown()                                   # 画笔抬起
#             turtle.write("{} 次时 PI 的值是 {:.4f}".format(i, pi), font=(" 宋体",␣
 ↪18, "normal"))
#     turtle.hideturtle()  # 隐藏光标
#     turtle.update()      # 刷新
#     turtle.done()        # 结束绘制


if __name__ == '__main__':
    type_of_pi = input()              # 接收用户输入的字符串
    cal_pi = type_judge(type_of_pi)   # 调用判断类型的函数
    print(cal_pi)                     # 输出函数运行结果
```

拉马努金法

1

3.1415926535897936

11. 随机密码生成

[ ]:
```
import random
```

```
def genpwd(length):
    a=random.randint(10**(length-1),(10**length-1))
    return a


length = eval(input())
random.seed(17)
for i in range(3):
    print(genpwd(length))
```

12. 阶乘累加求和

[ ]:
```
def multi(n):
    if n==1:
        return 1
    else:
        return n*multi(n-1)

n=eval(input())
s=0
for i in range(1,n+1):
    s += multi(i)

print(s)
```

13. 猴子吃桃 II

[ ]:
```
def peach(n):
    ######################
    # 补充你的代码
    if n==10:
        return 1
    else:
        return (peach(n+1)+1)*2

    ######################

for i in range(10,0,-1):
    print("第{}天有{}只桃子".format(i,peach(i)))
```