实验报告 3

WA2214014 杨跃浙 人工智能 2 班

实验内容：

实验内容 1-选择与循环

1. 编写程序，模拟蒙特卡罗计算圆周率近似值的方法，输入掷飞镖次数，然后输出圆周率近似值。观察实验结果，理解实验结果随着模拟次数增多越来越接近圆周率的原因。

2.使用集合实现筛选法求素数。编写程序，输入一个大于 2 的自然数，然后输出小于该数字的所有素数组成的列表。

3.使用列表实现筛选法求素数。编写程序，输入一个大于 2 的自然数，然后输出小于该数字的所有素数组成的列表。

4. 使用 filter()函数统计列表中所有非素数。首先，使用列表推导式和标准库 random 生成一个包含 50 个介于 1~100 的随机整数的列表，然后编写函数 def isPrime(n)用来测试整数 n 是否为素数，接下来使用内置函数 filter()把函数 isPrime()作用到包含若干随机整数的列表 lst 上，最后程序输出一个列表，其中只包含列表 lst 中不是素数的那些整数。

5. 编写程序，使用枚举法验证 6174 猜想。

6. 编写程序模拟猜数游戏。程序运行时，系统在指定范围内生成一个随机数，然后提示用户进行猜测，并根据用户输入进行必要的提示（猜对了、太大了、太小了），如果猜对则提前结束程序，如果次数用完仍没有猜对，提示游戏结束并给出正确答案。

实验内容 2-字符串

1、密码字符由数字、小写字母、大写字母、标点符号组成，其安全强度可以分为强密码、中育、中低、弱密码。字符串中包含的字符种类越多，认为越安全。其中强密码表示字符串中同时含有数字、小写字母、大写字母、标点符号这 4 类字符，而弱密码密码表示字符串中仅包含 4 类字符串中的一种串，输出该字符串作为密码时的安全强度。编写程序，输入字符串，输出密码强度。

2、凯撒加密算法的原理是将明文中的每个英文字母替换为字母表中后面第 k 个字母。若超出字母表范围，循环使用字母表，比如字母 Z 的下一个是 A，字母 z

的下一个是 a。区分大小写。编写程序：输入一个字符串作为待加密的明文，然后输入一个整数作为凯撒加密算法的密钥,最后输出加密后的结果。

3、编写程序模拟打字练习成绩评定。给定原始内容 origin 和用户输入内容 userInput，要求长度不能超过 origin。判断对应位置字符是否一致，一致为正确，否则为错误。计算正确字符数量除以原始字符串长度得出成绩，百分制输出，保留 2 位小数。

4、 一封邮件中如果这样的符号：【、】、＊ 、-、/超过了一定的比例,则认为是垃圾邮件。编写程序,对给定的邮件内容进行分类，提示“垃圾邮件”或“正常邮件”。

实验原理：

1. 主要代码

实验内容 1-选择与循环

```
def test1():
    import random

    summation = 0
    cnt = int(input("Please enter the number of times you throw darts: "))
    for i in range(cnt):
        x = random.uniform(-1, 1)
        y = random.uniform(-1, 1)
        if x ** 2 + y ** 2 <= 1:
            summation += 1
    pi1 = summation / cnt * 4
    pi2 = 0
    for k in range(100000):
        pi2 += 16 ** (-k) * (4 / (8 * k + 1) - 2 / (8 * k + 4) - 1 / (8 * k
+ 5) - 1 / (8 * k + 6))
    print(f'The calculation result of Monte Carlo method: {round(pi1, 8)}')
    print(f'The approximate calculation formula for pi results in: {round(pi2,
8)}')
    print(f'The error: {round(abs(pi1 - pi2), 8)}')

def test2():
    n = int(input("Please enter a natural number greater than 2: "))
    numbers = set(range(2, n + 1))
    prime_numbers = set()
    while numbers:
```

```
            current = min(numbers)
            prime_numbers.add(current)
            numbers -= set(range(current, n + 1, current))
        print("A set composed of all prime numbers less than n: ", prime_numbers)

def test3():
        n = int(input("Please enter a natural number greater than 2: "))
        numbers = list(range(2, n + 1))
        prime_numbers = []
        while numbers:
            current = min(numbers)
            prime_numbers.append(current)
            numbers=[i for i in numbers if i not in list(range(current, n + 1,
current))]
        print("A list composed of all prime numbers less than n: ", prime_numbers)


def test4():
        import random

        def isPrime(n):
            if n <= 1:
                return False
            for i in range(2, n):
                if n % i == 0:
                    return False
            return True

        lst = [random.randint(1, 100) for _ in range(50)]
        non_prime_numbers = list(filter(lambda x: not isPrime(x), lst))
        print(non_prime_numbers)

def test5():
        import itertools
        s = list(itertools.combinations(range(10), 4))
        flag = False

        for i in s:
            snum = ''.join(list(map(lambda x: str(x), i)))
            flag = False
```

```
            for j in range(7):
                l = sorted(snum)
                min = int(''.join(l))
                max = int(''.join(reversed(l)))
                if (max - min == 6174):
                    flag = True
                    break
                else:
                    snum = str(max - min)
            if (flag == False):
                break

        if (flag):
            print('6174 conjecture is true')
        else:
            print('6174 conjecture is false')

    def test6():
        import random

        def guess_number(min_num, max_num, max_tries):
            target = random.randint(min_num, max_num)
            print(f"请在{min_num}到{max_num}之间猜一个数字，你有{max_tries}
    次机会。")

            for i in range(max_tries):
                guess = int(input("请输入你猜测的数字："))

                if guess == target:
                    print("恭喜你猜对了！")
                    return
                elif guess < target:
                    print("太小了！")
                else:
                    print("太大了！")

            print(f"很遗憾，你没有猜对。正确答案是{target}。")

        min_num = 1
        max_num = 100
```

```
    max_tries = 7
    guess_number(min_num, max_num, max_tries)

if __name__=='__main__':
    test1()
    test2()
    test3()
    test4()
    test5()
    test6()
```

实验内容 2-字符串

```
def test1():
    import string

    def password_strength(password):
        has_digit=0
        has_lower=0
        has_upper=0
        has_punctuation=0

        for char in password:
            if char.isdigit():
                has_digit = 1
            elif char.islower():
                has_lower = 1
            elif char.isupper():
                has_upper = 1
            elif char in string.punctuation:
                has_punctuation = 1

        ans=has_digit+has_punctuation+has_lower+has_upper
        if ans==4:
            return "强密码"
        elif ans==3:
            return "中育密码"
        elif ans==2:
            return "中弱密码"
        else:
            return "弱密码"
```

```
    password = input("请输入密码：")
    print("密码强度为：", password_strength(password))

def test2():
    def caesar_encrypt(plaintext, key):
        ciphertext = ""
        for char in plaintext:
            if char.isalpha():
                shift = key % 26
                if char.islower():
                    ciphertext += chr((ord(char) - ord('a') + shift) % 26 +
ord('a'))
                else:
                    ciphertext += chr((ord(char) - ord('A') + shift) % 26 +
ord('A'))
            else:
                ciphertext += char
        return ciphertext

    plaintext = input("请输入待加密的明文：")
    key = int(input("请输入凯撒加密算法的密钥："))
    ciphertext = caesar_encrypt(plaintext, key)
    print("加密后的结果为：", ciphertext)

def test3():
    def typing_practice(origin, userInput):
        if len(userInput) > len(origin):
            return "错误：用户输入内容长度超过原始内容"

        correct_count = 0
        for i in range(len(userInput)):
            if origin[i] == userInput[i]:
                correct_count += 1

        score = correct_count / len(origin) * 100
        return f"成绩：{score:.2f}%"

    origin = "Positron Emission Tomography is a nuclear medicine imaging
technique used to observe specific aspects of biological activity in human or
```

```
animal bodies."
    userInput=input()
    print(typing_practice(origin, userInput))

def test4():
    def classify_email(content):
        symbols = ['[', ']', '*', '—', '/']
        total_count = len(content)
        symbol_count = sum([content.count(symbol) for symbol in symbols])
        ratio = symbol_count / total_count
        return(ratio)

    ratio=classify_email(input())
    if ratio > 0.2:
        print("垃圾邮件")
    else:
        print("正常邮件")

if __name__=='__main__':
    test1()
    test2()
    test3()
    test4()
```

2. 运行结果

实验内容 1-选择与循环

```
C:\Users\yangy\AppData\Local\Programs\Python\Python38\python.exe E:/Python/Project/Tset/Test3/Test1.py
Please enter the number of times you throw darts: 10000
The calculation result of Monte Carlo method: 3.1224
The approximate calculation formula for pi results in: 3.14159265
The error: 0.01919265
Please enter a natural number greater than 2: 20
A set composed of all prime numbers less than n:  {2, 3, 5, 7, 11, 13, 17, 19}
Please enter a natural number greater than 2: 30
A list composed of all prime numbers less than n:  [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
[88, 21, 68, 76, 38, 28, 82, 76, 32, 22, 81, 50, 52, 80, 60, 28, 92, 52, 55, 98, 66, 30, 12, 78, 54, 84, 72, 82, 52, 10, 39, 74, 38, 48, 16, 85, 52]
6174 conjecture is true
请在1到100之间猜一个数字，你有7次机会。
请输入你猜测的数字：50
太小了！
请输入你猜测的数字：70
太小了！
请输入你猜测的数字：90
太大了！
请输入你猜测的数字：80
太大了！
请输入你猜测的数字：75
恭喜你猜对了！

进程已结束,退出代码0
```

实验内容 2-字符串

```
C:\Users\yangy\AppData\Local\Programs\Python\Python38\python.exe E:/Python/Project/Tset/Test3/Test2.py
请输入密码：123456
密码强度为： 弱密码
请输入待加密的明文：I eat an apple every day.
请输入凯撒加密算法的密钥：3
加密后的结果为： L hdw dq dssoh hyhub gdb.
Positron Emission Tomography is a nuclear medicine iamge technique used to observe specifiv aspects of biological activity in human or animal bodies.
成绩：37.75%
[] I don't think,,,,
正常邮件

进程已结束,退出代码0
```

小结与讨论：

实验内容 1-选择与循环

题 1 中蒙特卡罗方法计算圆周率的原理基于随机抽样。具体来说，假设在一个边长为 1 的正方形内部画一个相切的圆，那么这两个图形的面积之比正好是圆周率π与 4 的比例。为了计算这个比例，我们可以进行多次随机抽样。
例如，我们可以生成大量的随机点，这些点的 x 和 y 坐标都在(-1, 1)范围内。然后，我们统计这些点中有多少个点的 x 和 y 坐标的平方和的平方根小于或等于 1，也就是落在了正方形内的圆内。根据统计学原理，随着抽样次数的增加，落在圆内的点数与总抽样点数之比会越来越接近圆与正方形面积之比，即π/4。因此，通过这个方法，我们可以得到一个越来越接近真正π值的近似值。
题 2 与题 3 中，筛选法求素数，也被称为埃拉托色尼筛法，是一种用于寻找所有小于 N 的素数的方法。它的基本思想是从 2（素数是大于 1 的自然数）开始，将某一范围内的正整数从小到大按顺序排列，然后逐步筛掉非素数，最后留下的就是素数。题 2 和题 3 分别利用了集合和列表的特性用 Python 实现了筛选法求素数。
题 4 中首先，需要编写一个函数 isPrime(n)来判断整数 n 是否为素数。然后，使用列表推导式和 random 库生成一个包含 50 个介于 1~100 的随机整数的列表。接着，使用内置函数 filter()把函数 isPrime()作用到包含若干随机整数的列表 lst 上。最后，程序输出一个列表，其中只包含列表 lst 中不是素数的那些整数。
题 5 中，6174 猜想指的是任给出四位数 k0,用它的四个数字由大到小重新排列成一个四位数 m,再减去它的反序数 rev(m),得出数 k1=m-rev(m),然后，继续对 k1 重复上述变换，得数 k2.如此进行下去，卡普耶卡发现，无论 k0 是多大的四位数，只要四个数字不全相同，最多进行 7 次上述变换，就会出现四位数 6174。通过编写程序验证发现他是正确的。
题 6 中，定义了一个名为 guess_number 的函数，接受三个参数：最小值、最大值和最大尝试次数。函数内部首先生成一个随机数作为目标数字，然后循环提示用户进行猜测，根据用户输入进行相应的提示。如果猜对则提前结束程序，如果次数用完仍没有猜对，提示游戏结束并给出正确答案。

实验内容 2-字符串

题 1 中首先需要判断输入的字符串中是否包含数字、小写字母、大写字母和标点符号这四种字符，然后根据包含的字符种类来判断密码强度。

题 2 中需要编写一个函数，接收两个参数，一个是待加密的明文字符串，另一个

是凯撒加密算法的密钥。然后遍历明文中的每个字符，判断其是否为英文字母，如果是，则根据密钥进行替换，如果不是，则保持不变。最后将替换后的字符拼接成新的字符串并返回。

题 3 中需要判断用户输入内容的长度是否超过原始内容，如果超过则输出错误信息。然后遍历原始内容和用户输入内容，比较对应位置的字符是否一致，统计正确字符数量。最后计算成绩并输出。

题 4 中我们需要定义一个函数，该函数接受一个字符串作为输入，这个字符串代表邮件的内容。然后，我们需要计算输入字符串中特定符号（[、]、*、-、/）的比例。可以通过遍历字符串并计数这些符号来实现这一点。接下来，我们需要判断这个比例是否超过了一定的阈值。如果超过了阈值，我们就认为这是一个垃圾邮件，否则认为这是一个正常邮件。