实验 2. 分支和迭代

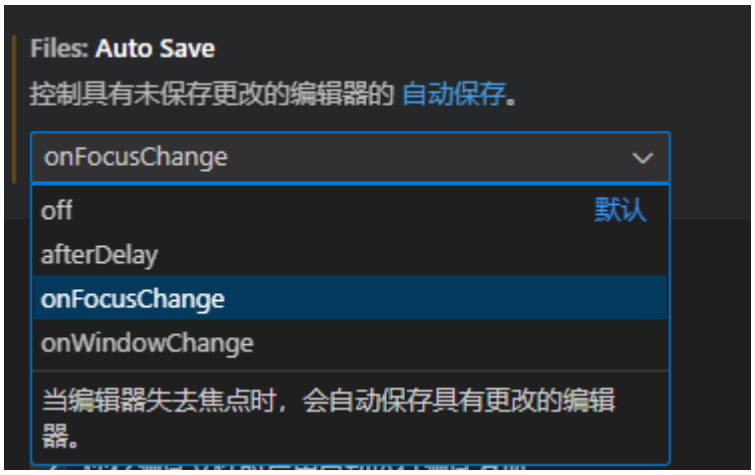
一、使用 VSCode 配置实验环境

打开 VSCode，Ctrl+Shift+P，输入 Python: Select Interpreter 来选择合适的环境。本次实验使用默认环境即可，若使用电三楼 519 机房，可以使用默认环境或 conda 的 basic 虚拟环境。注意运行代码之前要先保存，否则执行的上次保存的代码。

另外推荐的设置：

1、自动保存

左下角选择设置，搜索 auto save，选择自己的偏好。



二、实验内容

人工智能
数学原理与算法

1. 考拉兹猜想

定义一个从给定正整数 n 构建一个整数序列的过程如下。开始时序列只包含 n。如果序列的最后一个数 m 不为 1 则根据 m 的奇偶性向序列追加一个数。如果 m 是偶数，则追加 $m/2$，否则追加 $3 \times m + 1$。考拉兹猜想 (Collatz conjecture) 认为从任意正整数构建的序列都会以 1 终止。

编写程序读取用户输入的正整数 n，然后在 while 循环中输出一个以 1 终止的整数序列。输出的序列显示在一行，相邻的数之间用空格分隔。例如用户输入 17 得到的输出序列是“17 52 26 13 40 20 10 5 16 8 4 2 1”。

在本题中，我们需要执行：
1、使用 input()函数读入 n。input 函数的返回值是一个 str，需要用 int(input())来转换成整数。
2、检查用户输入的 n 是否小于或等于 0。
3、输出初始的 n
4、进入循环，循环的退出条件是 n 等于 1
5、在循环中，判断 n 的奇偶性，做出不同的操作。
注意：默认情况下，print() 会在输出后换行，而我们用 print(n, end=" ")表示输出后不换行，而是加一个空格。这样每次迭代的结果会显示在同一行。

2. 字符串加密

人工智能
数学原理与算法

编写程序实现基于偏移量的字符串加密。加密的过程是对原字符串中的每个字符对应的 Unicode 值加上一个偏移量，然后将得到的 Unicode 值映射到该字符对应的加密字符。

用户输入一个不小于-15 的非零整数和一个由大小写字母或数字组成的字符串，程序生成并输出加密得到的字符串。例如用户输入 10 和字符串“Attack at 1600”得到的加密字符串是“K~~kmu*k~*;@::”。

在本题中，我们需要执行：
1、读入偏移量 n 并检查是否在有效范围内。
2、读入待加密的字符串s。
3、遍历字符串中的每个字符，使用 ord(char)获取其 Unicode 值，加上偏移量 n，然后用 chr(int) 转换为新字符。
4、将加密后的字符拼接成字符串并输出。

3. 推导式转换为 for 语句

将程序 3.15 中的所有推导式转换为 for 语句。

程序 3.15 在 QQ 群里的 txt 文档中给出：

```
nums = {25, 18, 91, 365, 12, 78, 59}

multiplier_of_3 = [n for n in nums if n % 3 == 0]

print(multiplier_of_3)

square_of_odds = {n*n for n in nums if n % 2 == 1}

print(square_of_odds)



s = [25, 18, 91, 365, 12, 78, 59, 18, 91]

sr = {n:n%3 for n in set(s)}

print(sr)

tr = {n:r for (n,r) in sr.items() if r==0}

print(tr)
```

题目提供了 4 个推导式，包括列表推导式、集合推导式和字典推导式。我们要将每个推导式转换为等价的 for 循环形式。
注意每个推导式的返回值类型，向正确类型的空初始值中插入内容。列表插入元素的方式是 list.append(n)，集合插入元素的方式是 set.add(n)，字典的插入方式是 sr[n] = m。

1. 二分查找

编写一个程序使用二分法查找给定的包含若干整数的列表 s 中是否存在给定的整数 k。使用二分查找的前提是列表已按照从小到大的顺序排序。为此，程序需要先判断 s 是否已经排好序。若未排好序，则需调用 qsort 函数进行排序并输出排序结果。

程序 4.21 已列出了部分代码，需要实现函数 is_sorted 和递归函数 binary_search。binary_search 在列表 s 的索引值属于闭区间 [low,high] 的元素中查找 k，若找到则返回 k 的索引值，否则返回-1。

程序 4.21 在 QQ 群里的 txt 文档中给出：

```
def is_sorted(s):

#待完成

def qsort(s):

    if len(s) <= 1: return s

    s_less = []; s_greater = []; s_equal = []

    for k in s:

        if k < s[0]:

            s_less.append(k)

        elif k > s[0]:

            s_greater.append(k)

        else:

            s_equal.append(k)

    return qsort(s_less) + s_equal + qsort(s_greater)
```

```
def binary_search(s, low, high, k):

#待完成

s = [5, 6, 21, 32, 51, 60, 67, 73, 77, 99]

if not is_sorted(s):

    s = qsort(s)

    print(s)



print(binary_search(s, 0, len(s) - 1, 5))

print(binary_search(s, 0, len(s) - 1, 31))

print(binary_search(s, 0, len(s) - 1, 99))

print(binary_search(s, 0, len(s) - 1, 64))

print(binary_search(s, 0, len(s) - 1, 51))
```

本实验需要完成 is_sorted 和 binary_search 两个函数。

is_sorted 函数可以使用循环遍历数组，逐个与前一个数值比较，查找是否有不符合顺序的位置，如果有则返回 False，如果都满足有序就返回 True。

本次实验 binary_search 函数请使用递归写法，先判断[low, high]闭区间中点的值，与期望查找的位置 k 做比较，根据中点值与 k 的大小关系，调用 binary_search(s, low, mid-1, k)或者 binary_search(s, mid+1, high, k)，或直接返回 mid 位置。注意递归函数需要推出条件，比如数组的长度小于 2。注意若[low, high]区间长度是 2，即 high=low+1，递归查找的[low, mid-1]区间长度可能是负的，可以设置长度小于等于 2 作为推出条件，或特判长度为负的情况。若无法查找到 k，则返回-1。

本次实验我们不考虑数组有重复的值。