计算概论cheat sheet——计概加油！一．语法：

1. 输入输出类

（1）读取多行输入：

法一：应用循环 while True: try:

line = input() # 读取一行输入

print(f"你输入的内容是：{line}")#或者主要代码部分 except EOFError:

break # 捕获到 EOFError 时退出循环

举例：快速堆猪（多行输入与辅助栈的应用结合例题）

a = []

m = []

while True: try:

s = input().split() if s[0] == "pop":

if a:

a.pop() if m:

m.pop() elif s[0] == "min":

if m:

print(m[-1])

else:

h = int(s[1]) a.append(h) if not m:

m.append(h) else:

k = m[-1]

m.append(min(k, h)) except EOFError:

break

法二：sys:

import sys# 使用 sys.stdin.read() 读取所有输入

data = sys.stdin.read() # 读取整个输入流

print(data)

(2)list(map(int,input().split()))注意map只是一个生成器，要用list承接，这里应用于输入全是整数如果不全是整数可以直接input.split() 返回的也是一个列表

注意如果输入中间没有间隔，不要加split，但这样也可以通过 matrix[i][j] 搜索到。

1. 浮点数的输入输出

float(input())

输出：

法一：f"{value:.nf}" #其中n是希望保留的小数位数百分数

print(f"Percentage: {percentage:.2%}") # 输出：85.00%

科学计数法：

print(f"Scientific notation: {large\_number:.2e}") # 输出：1.23e+06

1. 格式化输出：

基本模板：f"字符串 {表达式} 字符串"

print(f"My name is {name} and I am {age} years old.")

三元运算符：

print(f"Adult status: {'Yes' if age >= 18 else 'No'}")

1. 其他注意：

无空格输出： print(' '.join(map(str, minStep[i]))) #注意这里应该是字符串形式保护圈： maze.append([-1] + [int(\_) for \_ in input().split()] + [-1])

1. 从列表中删除元素：

法一：my\_list = [1, 2, 3, 4, 2, 5] my\_list.remove(2) # 删除第一个 2

法二：my\_list = [1, 2, 3, 4, 5]

removed\_element = my\_list.pop(2) # 删除索引为 2 的元素 (即 3)

1. 字典与集合的使用（包括lambda函数的创建）
2. 字典的使用：

# 创建字典

my\_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}

#通过键来搜索值

法一：print(my\_dict['name']) # 输出：Alice

法二：print(my\_dict.get('name')) # 输出：Alice print(my\_dict.get('address', 'Not Found')) # 输出：Not Found

#通过值来搜索键找到所有的键：

法一：keys = [key for key, value in my\_dict.items() if value == search\_value]法二：keys = list(filter(lambda key: my\_dict[key] == search\_value, my\_dict))找到第一个符合条件的键：

key = next((key for key, value in my\_dict.items() if value == search\_value), None)

#添加或更新元素（键值对）： my\_dict['age'] = 26 # 更新 my\_dict['country'] = 'USA' # 添加

#向字典中某一个键下添加元素：

my\_dict = {'key1': [1, 2, 3], 'key2': [4, 5]} my\_dict['key1'].append(4)

#删除键值对

法一：del my\_dict['city']

# 删除 'city' 键值对

法二：age = my\_dict.pop('age')

print(age) # 输出：26

#遍历字典：

# 遍历键

for key in my\_dict:

# 遍历值

for value in my\_dict.values(): print(value)

# 遍历键值对

for key, value in my\_dict.items(): print(f"{key}: {value}")

#字典推导式举例：

numbers = [1, 2, 3, 4, 5]

squared\_dict = {n: n\*\*2 for n in numbers} print(squared\_dict)

#字典排序：

sorted\_dict = dict(sorted(my\_dict.items(), key=lambda x: x[1], reverse=True))

1. 集合的使用：

# 创建集合

my\_set = {1, 2, 3, 4, 5}

another\_set = set([3, 4, 5, 6, 7])

#注意：set() 用来创建集合时，它接受一个可迭代对象（如列表、元组、字符串等），因而这里set() 会自动从列表中提取元素并创建集合，而不能直接set(3, 4, 5, 6, 7)，因为set()括号里只可以有一个参数，而{}则不同。

# 添加元素

my\_set.add(6)

# 删除元素（不存在元素可抛出错误）

my\_set.remove(2)

# 删除不存在的元素，不会抛出错误

my\_set.discard(10)

1. lambda函数的使用（主要是在字典排序中）

基本模板：lambda arguments: expression #参数：对参数进行的操作

在字典排序中：

sorted\_dict = sorted(my\_dict.items(), key=lambda x: x[1]) #按值升序排序，注意sorted得到的是一个列表!

#如果想要降序并转化为字典格式如下：

sorted\_dict = dict(sorted(my\_dict.items(), key=lambda x: x[1], reverse=True))

与map结合：

# 对列表中的每个元素进行平方操作

squared\_numbers = list(map(lambda x: x \*\* 2, numbers))

1. 排序的几种方法与适用范围（语法方面）：
2. sorted() **返回一个新的排序后的列表**，原始的可迭代对象不被修改。可以作用于任何可迭代对象，



key

包括列表、元组、字典等。支持自定义排序规则（通过

或需要排序结果为列表的场景。

参数）。用于需要保持原始数据不变的情况，

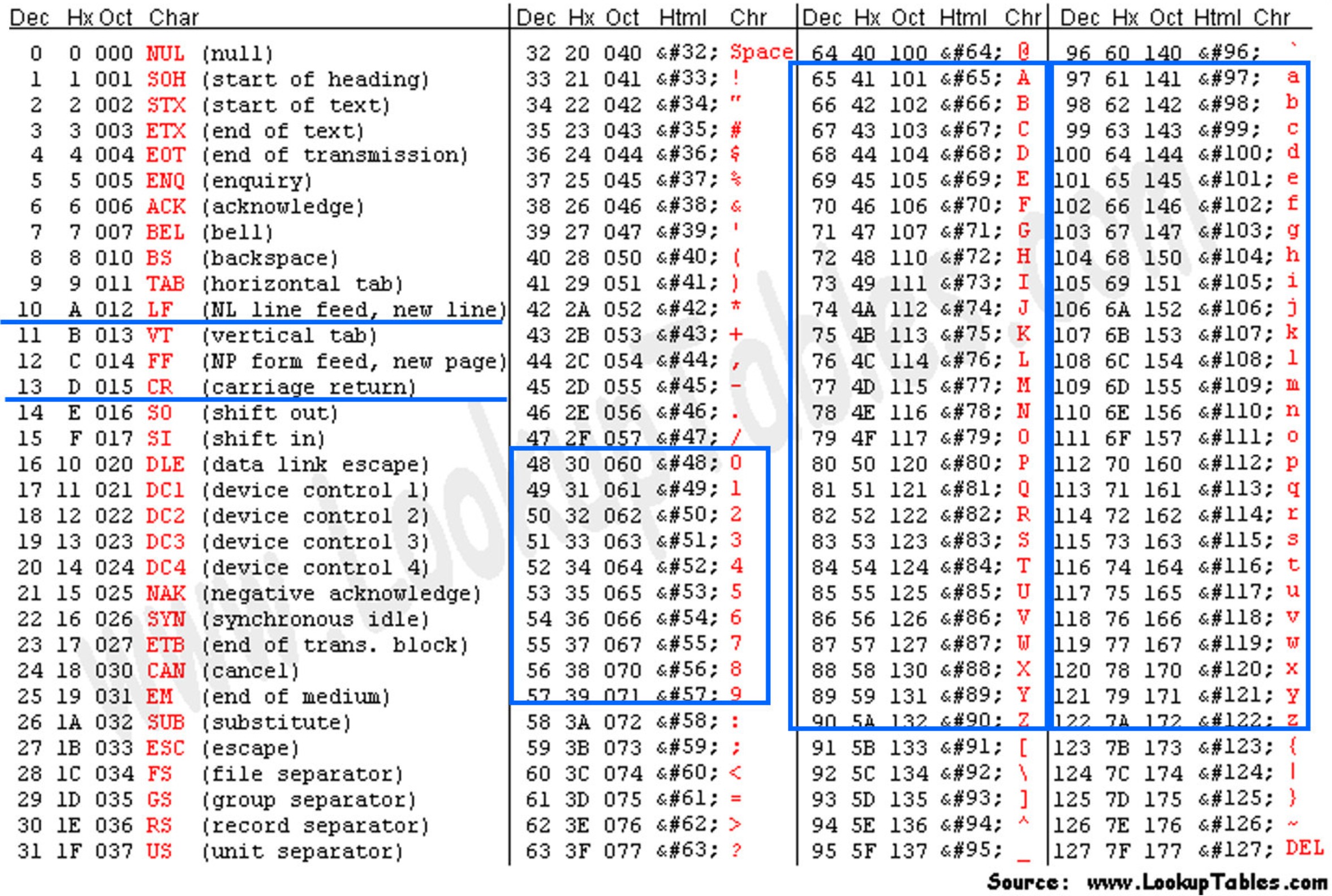
如按长度排序：（字典中的用法见上一条）

words = ["banana", "apple", "pear", "cherry"] sorted\_words = sorted(words, key=len)

对列表中的数组按照数组的第一个数字排序：

sorted\_dist=sorted(dist,key=lambda x:x[0])

1. .sort() 是 **列表对象的方法**，只能对 **列表** 进行排序，原地修改列表，即 排序会直接修改原列表，返回值为 None 。适用于不需要保留原列表的情况，或者希望节省内存的场景。
2. ASCII表及其用法



调用语法：

# 获取字符的 ASCII 码——ord() ascii\_value = ord('A') print(ascii\_value) # 输出：65

# 获取 ASCII 码对应的字符——chr() char = chr(65)

print(char) # 输出：A

1. enumerate:（返回枚举对象）

基本语法：enumerate(iterable, start=0)

iterable：这是想要遍历的可迭代对象（如列表、元组、字符串等）。 start：可选，指定索引从哪里开始，默认从 0 开始。

返回元组，每个元组由两个元素组成：第一个是索引，第二个是对应的值。

举例1：单纯返回索引和元素

fruits = ['apple', 'banana', 'cherry'] for index, fruit in enumerate(fruits):

print(f"Index: {index}, Fruit: {fruit}")

举例2：与if搭配实现查找：

fruits = ['apple', 'banana', 'cherry', 'orange'] for index, fruit in enumerate(fruits):

if 'a' in fruit:

print(f"Found fruit with 'a' at index {index}: {fruit}")