day3

一、今日内容

a. 数据类型 -- 列表
b. 数据类型 -- 元组
c. 数据类型 -- 字典
d. python 函数
    i. 函数定义
    ii. 函数调用
    iii. 函数返回值
    iv. 函数参数
    v. 局部变量以及全局变量
    vi. 递归函数
        1. 概念
        2. 定义
        3. 时间复杂度
        4. 面试中的递归

二、昨日内容

a. 逻辑语句 -- 循环语句
    i. while
    ii. for: for 临时变量 in 可迭代对象:
    iii. 循环语句控制关键字: break continue
b. 数据类型--- 字符串
    i. 定义: 引号之间的内容就是所谓的字符串
    ii. 访问:
        1. 下标: python支持负下标
        2. 切片: var[起始位置: 终止位置: 步长]
    iii. 字符串的基础操作
        1. 运算符: + * == !=
        2. 转换: upper, lower
        3. 分割家族: split splitlines
        4. 查询家族: find rfind index rindex
c. 数据类型 --- 列表
    i. 概念: 容器: [int, bool, string, list, tuple, dict]
    ii. 访问:
        1. 下标
        2. 切片

iii. 列表的增删改

1. 增加: append insert extend
2. 删除: pop del remove
3. 修改: list[pos] = value

三、数据类型 --- 列表

3.1 列表的查找

```
# 列表的查找
"""
in :  判断元素是否在容器内, 返回值类型是 bool  True or False
格式:   xxx  in  list; print("貂蝉 in home :", "貂蝉" in home )

not in: 判断元素是否不在容器内, 返回值类型 bool
格式:   xxx not in  list: print("貂蝉 not in home", "貂蝉" not in home)

count:  统计 element 在列表中出现的个数
格式:  list.count(element): print("job.count(洗浴中心-刮痧师傅)", job.count("洗浴中心-刮痧师傅"))

index: 使用方式和字符串一致, 返回值的元素在容器中的下标,如果元素不存在容器中,则会报错.
格式:  list.index(element):
"""
print("home.index(狄仁杰) :" , home.index("狄仁杰11111"))

```

3.2 排序

```
# 排序: 冒泡排序  快排
# month = [1, 3, 5, 7, 8, 10, 12, 4, 6, 9, 11, 2]

month = [1, 2, 4, 3, 5, 6, 7, 8, 9, 10, 11, 12]

# 使用冒泡排序, 从小到大排序


i = 0
count = 0
while i < len(month) - 1:
```

```
    flag = 0
    j = 0
    while j < len(month) - 1 - i:

        #  前面值 >  后面值
        if month[j] > month[j + 1]:
            flag = 1
            month[j], month[j + 1] = month[j + 1], month[j]
        count += 1
        j += 1

    if flag == 0:
        break
    i+= 1
print(month)
print(count)
```

四、数据类型 -- 元组

4.1 概念

元组是一个容器，这个容器是不可以进行修改的。 (int, bool, list, string ,list, tuple)

i. 可修改：在原有的数据之上进行修改 list

ii. 不可修改： 在原有的数据之上不可以进行修改 string

4.2 访问

i. 下标访问

1. 从左到右： 下标从0 开始
2. 从右到左： 下标从-1开始

ii. 切片访问 tuple[初始位置： 终止位置：步长]

4.3 查找

i. in: 查找元素是否在元组内

ii. not in : 查找元素是否不在元组内

iii. index：查找元素所在的位置

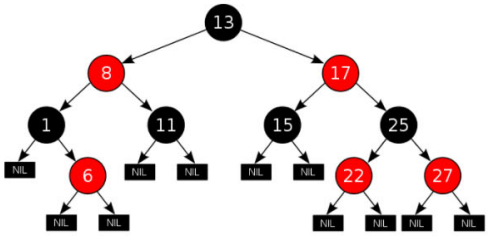
iv. count: 统计元素在容器中的出现的次数

五、数据类型 -- 字典

5.1 概念

字典是一个容器， 使用{key: value，key : value}. 底层原理，使用的红黑树。key值必须是不可以修改的类型。

5.2 红黑树



<1>结点是红色或黑色
<2>根结点始终是黑色
<3>叶子结点（NIL 结点）都是黑色
<4>红色结点的两个直接孩子结点都是黑色（即从叶子到根的所有路径上不存在两个连续的红色结点）
<5>从任一结点到每个叶子的所有简单路径都包含相同数目的黑色结点

5.3 字典的访问

```
# 字典的访问
egg = {"name": "卤蛋", "age": 5}

# 字典的访问:  字典是通过 key值进行访问的。
# 通过[] 配合 key值进行访问，如果key值不存在，则会报错
# print("my name is %s, age is %d. father is %s" %(egg["name"], egg["age"],egg["father"]))

# 通过 get函数配合 key值进行访问。 如果key值不存在,则返回None : None类似于C语言中的NULL
print("my name is %s, age is %d, father is %s."% (egg.get("name"), egg["age"], egg.get("father")))
```

5.4 字典的增删改查

```
# 字典的访问
egg = {"name": "卤蛋", "age": 5}

# 字典的增加: 格式 dict[new_key] = value
egg["father"] = "幼儿园院长"
egg["mother"] = "院长夫人"
egg["aunt"] = "班主任"

# 字典的修改: 格式:  dict[old_key] = value
egg["father"] = "大公鸡"

# 字典的修改:  格式:  del dict[key]: 字典的修改，通过key值进行修改，如果没有给出key,则会删除整
# del egg["mother"]
```

```
# del egg["aunt"]

# 字典的清空:  格式:  dict.clear()
# egg.clear()

# 字典的查找。   in   not in 配合查找
"""
len(): 返回字典的键值对儿个数 使用方式:  len(dict)
keys(): 返回字典中的所有的key值,使用方式: dict.keys()
values(): 返回字典中的所有的value值,  使用方式:  dict.values()
items(): 返回字典中的键值对儿 使用方式: dict.items()
"""
print("len (egg): ", len(egg))
print("egg.keys() : ", egg.keys())
print("name in egg.keys(): ", "name" in egg.keys())
print("egg.values(): ", egg.values())
print("egg.items(): ", egg.items())

# items应用在遍历过程中

for key, value in egg.items():
    print(key, "----", value)

```

六、函数

6.1 概念

对某些功能的一个封装。 本质: 方便代码的复用。要求:高内聚低耦合。

6.2 函数定义

```
格式:
def 函数名(形式参数列表):
(tab)函数体
    return
demo:

函数调用:
函数名(实际参数列表)

# 定义一个函数,计算两个数字的加法
```

```python
def add(x, y): # python的形式参数是不需要类型的

    sum = x + y
    return sum


# 调用函数  函数名(实际参数列表)
result = add(10, 20)
print("result :  ", result)

tmp = add("hello", " world")
print("tmp : ", tmp)

question: 形式参数与实际参数之间的关系是什么?
answer: 形式参数是实际参数的一份临时拷贝,形式参数的改变不会影响到实际参数
```

6.3函数的返回值

```python
# python中的函数,支持返回多个值
def operator(x, y):

    add = x + y
    sub = x - y
    mul = x * y
    div = x / y

    # 多个返回值之间,使用',' 隔开
    return add, sub, mul, div

# 函数返回多个值的接收结果

# 1. 使用与返回值个数相同的变量接收
# add, sub, mul, div = operator(100, 200)
#
# print("add : ", add)
# print("sub : ", sub)
# print("mul : ", mul)
# print("div : ", div)

# 2. 使用一个变量接收，会将所有的值打包成元组处理。
result = operator(100, 200)
```

```
print("result : ", result)
```

6.4 局部变量和全局变量

i. 局部变量： 函数内部定义的变量。

ii. 全局变量：函数外部定义的变量

iii. global: 在函数内部声明,使用的变量是全局变量。

```
money = 100

def modify():

    # global 关键字
    global money

    # 局部变量：python中的变量是不需要提前声明
    money = 200

modify()

print("money : ", money)
```

6.5 递归函数

6.5.1 概念

函数调用函数本身(自己调用自己的函数)。思路：将一件单的事情，转换为多种执行方式相同的小问题。

要求：

a. 需要由一个终止条件

b. 每递归一次，都要向结束条件靠近。

6.5.2 课堂练习 -- 阶乘

```
# target: 计算某个数的阶乘 5!  5 * 4 * 3 * 2 * 1 = 5 * 4 ! ~  n * (n-1)

def jiec(n):

    # 递归函数
```

```python
    if n == 1:

        return 1

    return n * jiec(n - 1)

print(jiec(5))
```

6.5.3 时间复杂度

递归函数的事件复杂度： 函数复杂度 * 递归的次数

6.5.4 递归函数的应用场景

1. 想得到就用，想不到就不用
2. 对时间复杂度和空间复杂度不会有太大影响的时候

6.5.5 面试 -- 青蛙跳台阶问题

七、学生管理系统

```python
"""
学生管理系统:

描述学生信息:
student = {"name" : name, "age" : age, "score" : score};

组织信息
python: 列表组织数据
class_info = [{"name" : name, "age" : age, "score" : score},{"name" : name, "age" : age,

"""

# 定义学生班级信息
class_info = []


def print_menu():
    print("----------------------------")
    print("      学生管理系统 V1.0")
    print(" 1:添加学生")
    print(" 2:删除学生")
```

```python
    print(" 3:修改学生")
    print(" 4:查询学生")
    print(" 5:显示所有学生")
    print(" 6:退出系统")
    print("--------------------------")


def add_student():

    # 声明使用全局变量 class_info
    global class_info

    # 输入学生信息
    name = input("请输入添加学生的姓名: >>>")
    age = int(input("请输入添加学生的年龄: >>>"))
    score = int(input("请输入添加学生的成绩: >>>"))

    # 判断输入信息
    for student in class_info:

        if student["name"] == name:
            print("您输入的学生已经存在! ")
            return -1

    if 0 >= age:
        print("您输入的年龄有误! ")
        return -2

    if 0 > score or score > 101:
        print("您输入的成绩有误!")
        return -3

    # 描述学生信息
    student = {
        "name": name,
        "age": age,
        "score": score
    }

    # 将学生信息添加到班级信息中
    class_info.append(student)
    print("恭喜您,添加成功")
    return 0

```

```python

def del_student():

    # 声明使用全局变量class_info
    global class_info

    # 输入需要删除的学生姓名
    name = input("请输入需要删除的学生姓名: >>>")

    # 寻找学生
    for student in class_info:

        if student["name"] == name:
            # 找到需要删除的学生
            class_info.remove(student)
            print("删除成功! ")
            return 0

    print("您输入的学生不存在")
    return -1



def main():

    while True:
        # 打印菜单
        print_menu()

        # 输入自己的选择
        choose = int(input("请输入您的选择: >>>"))

        if choose == 1:
            add_student()
            print(class_info)

        elif choose == 2:
            del_student()
            print(class_info)
            pass


main()

```

```
110
```