python基础5：进阶

python基础5：进阶

一、强化和进阶

1.【重点】组包和拆包

1.2. 【重点】组包和拆包的应用

2. 【理解】引用

2. 2 【理解】引用指向改变

2.3. 【记忆】函数传参是引用传递

3.【记忆】可变类型与不可变类型

4.【重点】range

5.【重点】列表推导式

6.【记忆】匿名函数

7.【了解】递归函数

8.【记忆】enumerate和del

二、【应用】学生名片管理系统

1.需求分析

2.主页面逻辑:

3.菜单实现

4.添加学生信息:

5.显示所有学生信息:

6.查询某个学生:

7. 修改某个学生

8. 删除某个学生

一、强化和进阶

1.【重点】组包和拆包

1. 组包
   - = 右边有多个数据时, 会自动包装为元组

```
result = 10, 20, 30
```

2. 拆包
   - **变量数量 = 容器长度**, 容器中的元素会一一对应赋值给变量

```
a, b, c = (10, 20, 30)
```

```
# 组包：= 右边有多个数据时，会自动包装为元组，多变一
# 拆包：如果 变量数量 = 容器长度，容器中的元素会一一对应赋值给变量，一变多，取出有用的信息


# 组包，1，2，3封装成元组再赋值，多变一
result = 1, 2, 3     # 第一步：1，2，3，包装成元组 (1,2,3) 第二步：赋值给result
```

```
print(result, type(result))  # (1, 2, 3)
<class 'tuple'>

# 拆包，一变多
my_tuple = (10, 20, 30)
a, b, c = my_tuple
print(a, b, c)

# 注意：变量数量 = 容器长度
# a, b = my_tuple  # ValueError: too many
values to unpack (expected 2)
# print(a, b)

# 列表拆包
my_list = [1, 2, 3]
x, y, z = my_list
print(x, y, z)

# 字典拆包
my_dict = {"name": "xiaoming", "age": 18}
x, y = my_dict       # 注意：字典的拆包，返回的
是key
print(x, y)
print(my_dict[x], my_dict[y])

# 字符串拆包
```

```
my_str = "hel"
x, y, z = my_str
print(x, y, z)
```

1.2. 【重点】组包和拆包的应用

```
# 交换变量的值
# 1. 借助辅助变量
a = 100
b = 200
print(a, b)

c = a
a = b
b = c
print(a, b)

# 2. 不借助辅助变量
a, b = b, a      # 第一步：b, a 组包 成 (b, a)
第二步：(b, a) 拆包 赋值给 a, b
print(a, b)
```

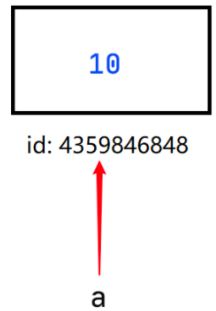
```python
# 函数可以同时返回多个数
def foo():
    return 1, 2, 3       # 1, 2, 3 组包 成 一个元组 (1,2,3)


# 函数调用
# 变量名 = 函数()
ret = foo()
print(ret, type(ret))   # (1, 2, 3) <class 'tuple'>


# 返回值直接做拆包
x, y, z = foo()
print(x, y, z)


# 3. 字典元素拆包
# 遍历字典，取出每一个item

my_dict = {"name": "小明", "age": 18, "sex": "男"}

# for item in my_dict.items():  # item是元组类型
```

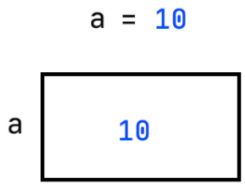
```
#     print(item, type(item))

for k, v in my_dict.items():     # 对元组进行拆包成 k, v
    print(k, v)
```

2. 【理解】引用

- 引用：是一个变量或值的另一个名字，又称**别名**
- 可以使用 **id函数** 查看变量的引用地址，**引用地址相等，说明指向同一个内存空间**
- 函数传参是引用传递

```
a = 10
print(id(a), id(10))
```





1）引用：理解为内存的别名，对于使用者而言，理解为上图即可
2）简单理解为 a 为 10 内存别名，操作a，就是操作10

```
"""
1. 引用：是一个变量或值的另一个名字，又称别名
2. id(变量名或值)：查看变量或值的引用地址
3. 引用地址相等，说明指向同一个内存空间
"""

a = 10  # a 是10的引用(别名), 10的别名是a.
print(a, id(a), id(10))

b = a   # b 是a 的引用(别名)
print(id(b), id(a))
```

2.2 【理解】引用指向改变

```
"""
1. 引用：是一个变量或值的另一个名字，又称别名
2. 赋值本质：给右边的变量或值，起一个别名
"""

a = 10
print(id(a), id(10))

b = a
print(id(a), id(b))

b = 20  # b 是20的引用，20的别名是b，引用地址改变.
print(id(b), id(20))
```

2.3. 【记忆】函数传参是引用传递

```
# 给函数传参是引用传递
# 带参数函数定义
def foo(num):    # num = a  num就是a的引用，函数传参就是引用传递
    print("foo函数中: ", id(num))


# 给函数传参，变量传参
a = 100
print("foo函数调用前: ", id(a))
foo(a)
print("foo函数调用后: ", id(a))
```

3.【记忆】可变类型与不可变类型

- 可变类型: `列表，字典，集合`
- 不可变类型: `数字类型（int, bool, float), 字符串，元组`

```
"""
可变类型：列表、字典、集合
不可变类型：数字类型(int, bool, float)、字符串、元组
可变类型：在地址不变的情况下，可以修改内容
不可变类型：在地址不变的情况下，不可修改内容
"""
```

```python
# 验证列表是可变类型

# 可变类型：在地址不变的情况下，可以修改内容
my_list = [1, 2, 3]
print(my_list, id(my_list))

my_list.append(4)
print(my_list, id(my_list))

print("=" * 50)

my_dict = {"name": "xiaoming"}
print(my_dict, id(my_dict))

my_dict['name'] = "小芳"
print(my_dict, id(my_dict))

print("=" * 50)

# 验证int是不可变类型
# 不可变类型：在地址不变的情况下，不可修改内容，(如果修改了,地址已经发生改变)
a = 10
print(a, id(a), id(10))
```

```
a = 20
print(a, id(a), id(20))

print("=" * 50)

my_tuple = (1, 2, 3)
print(my_tuple, id(my_tuple))
# my_tuple[0] = 20  # TypeError: 'tuple'
object does not support item assignment

my_tuple = (20, 2, 3)
print(my_tuple, id(my_tuple))
```

4.【重点】range

- range的使用：for 变量 in range(开始位置, 结束位置，步长)

```
# for 变量 in range(5), range(5)序列范围，使用和切片一样，但是以,隔开

# 1. 打印：0、1、2、3、4
# i = 0
# while i < 5:
#     print(i)
#     i += 1
```

```python
# for x in range(5):  # [0, 5)
for x in range(0, 5):
    print(x)


# 2. 1~100的累加
# i = 1
# _sum = 0
# while i < 101:
#     _sum += i
#     i += 1
# print("while: ", _sum)

# 2.1 定义辅助变量
_sum = 0
# 2.2 for 控制循环范围
for i in range(1, 101):
    # 2.3 累加
    _sum += i
# 2.4 在循环外面打印累加结果
print("for: ", _sum)


# 3. 验证步长，打印：0、2、4
for i in range(0, 5, 2):
```

```
    print(i)
```

5.【重点】列表推导式

- 能够使用列表推导式创建包含1-100之间元素的列表
  - 格式: `[计算公式 for 循环体 if 判断]`

```
# 列表推导式，  通过for添加列表元素的简洁写法

# 普通方法：遍历0~4范围的元素，这些元素添加到列表中
# 1. 空列表
new_list = []
# 2. range(5)遍历取数
for i in range(5):  # [0, 5)
    # 2.1 取出来的元素追加到列表
    new_list.append(i)
# 3. 循环外面，打印结果
print(new_list)


print('='*30)

# 通过列表推导式，实现上面的效果 [计算公式 for循环体]
```

```python
# 1. for i in range(5), 取出0，放在i变量中，i追加到列表
# 2. 循环下一步，取出2，放在i变量中，i追加到列表
# 重复，直到退出循环
new_list = [i for i in range(5)]
print(new_list)

print('='*30)


# 0~10之间数，偶数才添加到列表
# 普通方法实现
# 1. 空列表
new_list = []
# 2. range(11)遍历取数
for i in range(11):  # [0, 11)
    # 2.1 取出来的元素是偶数的话，追加到列表
    # 2.2 i % 2 == 0，i 对 2求余，结果为0，就是偶数
    if i % 2 == 0:
        new_list.append(i)
# 3. 循环外面，打印结果
print(new_list)


print('='*30)
```

```
# 列表推导式实现 [计算公式 for 循环 if 判断]
# [i for i in range(11) if i % 2 == 0]
# 1. for i in range(11)取第一个元素
# 2. if i % 2 == 0
# 3. 上面满足条件的i, 条件到列表
new_list = [i for i in range(11) if i % 2
== 0 ]
print(new_list)

new_list = [i*2 for i in range(11) if i % 2
== 0 ]
print(new_list)
```

6.【记忆】匿名函数

- 通过匿名函数编写简单的函数
  - 格式: `lambda [形参1, 形参2, ...] : 单行表达式 或者 函数调用`

```
# 给匿名函数起一个函数名字, 函数名字()就是调用函数
func = lambda: 1 + 1  # 给匿名函数起一个函数名
字叫func
ret = func()  # 返回值变量 = 函数名()
print(ret)
```

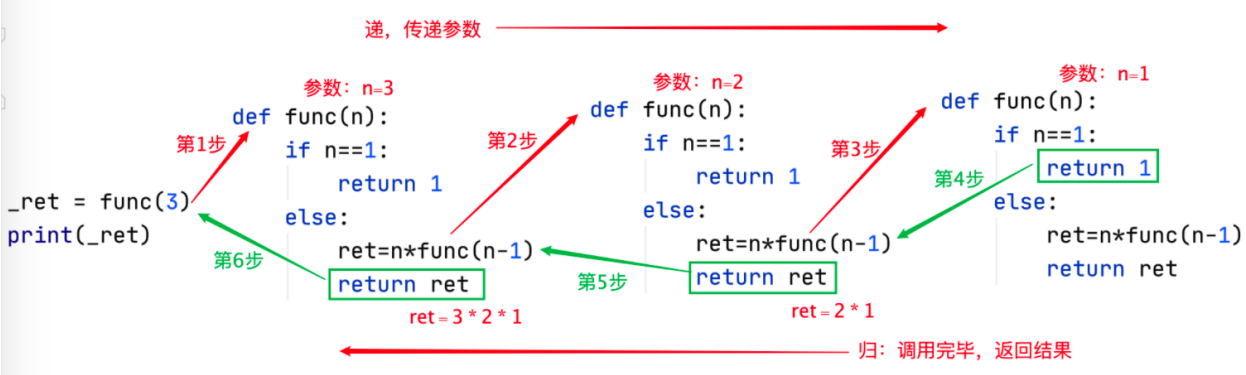
```python
# 匿名函数是简单普通函数的简洁写法
# 匿名函数没有函数名字

# 1. 无参有返回值
# 1.1 普通函数
# 函数定义
def foo():
    return 1 + 2


# 函数调用
ret = foo()
print(ret)

print("=" * 50)

# 1.2 匿名函数
# 定义：lambda ： 函数体
# 函数体就是返回值的内容，无需return
# lambda: 1 + 2

# 匿名函数调用
# a) 匿名函数整体就是函数名字，  函数名字()就是函数调用
ret = (lambda: 1 + 2)()
print(ret)
```

```python
# b) 给匿名函数起一个函数名字，函数名字()就是调用函数
func = lambda: 1 + 2
ret = func()
print(ret)


# 2. 有参有返回值
# 2.1 普通函数
# 函数定义
def foo(a, b):
    return a - b


# 函数调用
ret = foo(30, 10)
print(ret)

# 2.2 匿名函数
# a. 直接调用匿名函数
ret = (lambda a, b: a - b)(30, 10)
print(ret)

# b. 先给匿名函数起名，再调用
func = lambda a, b: a - b
ret = func(30, 10)
```

```python
print(ret)


# 扩展：匿名函数的应用
def foo(fn):
    ret = fn()
    print(ret)


foo(lambda: 1 + 2)
foo(lambda: 10 - 2)
```

7.【了解】递归函数

1. 递归函数特点
    - 函数自己调用自己
    - 一定有出口
2. 递归函数调用流程



```
"""
1. 函数递归：函数调用自己，了解即可，尽量通过画图，理解流程
2. 递归函数一般会在特定情况下 不再调用函数本身(出口)

阶乘:
1! = 1
2! = 2*1      = 2*1!
3! = 3*2*1    = 3*2!
4! = 4*3*2*1 = 4*3!
5! = 5*4*3*2*1=5*4!
n! = n*(n-1)……1=n*(n-1)!

# 1. 定义函数(参数)
# 2. 如果我是 1 ，直接返回 1
# 3. 否则，返回 n * 函数调用自己(n-1)
"""


def foo(n):
    if n == 1:  # 2.递归函数的出口
        return 1
    else:
        ret = n * foo(n - 1)     # 1.函数内调用本身(递归调用)
```

```
        return ret


ret = foo(5)
print(ret)
```

- 如果递归没有结束，会报错超出最大递归深度:

```
  [Previous line repeated 996 more times]
RecursionError: maximum recursion depth exceeded
```

8.【记忆】enumerate和del

1. 通过 for 配合 enumerate 遍历容器同时获取元素索引位置、元素
   - `for i, value in enumerate(容器):`
2. 通过del删除列表元素：`del 列表[索引]`

```
user_list = [{"name": "小明", "age": 20},
{"name": "小张", "age": 21}]

# 遍历列表，同时把索引位置能打印
# 普通方法实现
# 1. 定义索引位置变量
```

```python
i = 0
# 2. for遍历列表，打印：索引、元素
for item in user_list:
    print(i, item)
    # 3. 索引位置+1
    i += 1


print('=='*20)

# 通过enumerate方法实现
# enumerate(容器变量)：获取到：元素位置，元素
for i, item in enumerate(user_list):
    print(i, item)


print('=='*20)

# 通过del删除列表元素 del 列表[索引位置]
# 删除索引位置为0的元素
del user_list[0]
print(user_list)

del(user_list[0])
print(user_list)
```

二、【应用】学生名片管理系统

1.需求分析

```
"""
【应用】学生名片管理系统：存用户的基本信息（姓名、年龄、电话）
1．如何存一个用户的信息：
    字典存学生信息.
2．如果管理存储多个用户：
    列表存储多个学生信息.
3．增加学生：
    输入学生姓名,年龄,电话
        input()
        int(input())
        input()
    判断有没有添加过该学生,如果有不再添加,如果没有,再添加
    做成字典
    添加到列表中
4．查找显示所有的用户信息：
    for循环遍历列表获取学生信息
```

打印出来(索引 元素) <- enumerate(列表)

5. 查找某个用户
    获取输入: input()
    for循环遍历列表获取学生信息
        判断列表中是否存在该学生
        如果存在打印学生信息
    else
        查询的用户不存在

6. 修改用户
    获取输入: input()
    for循环遍历列表获取学生信息
        判断列表中是否存在该学生
        如果存在修改学习信息
    else
        查询的用户不存在

7. 删除用户
    获取输入: input()
    for循环遍历列表获取学生信息
        判断列表中是否存在该学生
        如果存在删除学习信息 del 列表[索引]
            7.1 找到需要删除用户所在位置的 num
            7.2 删除元素
    else

```
        查询的用户不存在
"""
```

2.主页面逻辑:

```
"""
# 1. 死循环
# 2. 用户输入数字
# 3. 条件选择
"""

# 1. 死循环
while True:
    # 2. 用户输入数字
    n = input("请输入功能数字: ")
    # 3. 条件选择
    if n == "1":
        print("添加学生")
    elif n == "2":
        print("显示所有学生")
    elif n == "3":
        print("显示一个学生")
    elif n == "4":
        print("修改一个学生")
    elif n == "5":
        print("删除一个学生")
```

```
    elif n == "6":
        break
    else:
        print("输入错误,请重新输入")
```

3.菜单实现

```
"""
1. 显示菜单
"""


def menu():
    """ 功能菜单 """
    print('=' * 10 + "学生名片管理系统" + '='
* 10)
    print("1. 添加学生")
    print("2. 查询所有学生")
    print("3. 查询某一个学生")
    print("4. 修改某一个学生")
    print("5. 删除某一个学生")
    print("6. 退出系统")
    print("=" * 34)
```

```python
# 定义主函数：主页面逻辑
def main():
    """ 主页面逻辑 """
    # 1. 死循环
    while True:
        menu()
        # 2. 用户输入数字
        num = int(input("请输入选择的功能：
"))

        # 3. 条件选择
        if num == 1:
            print("添加学生")
        elif num == 2:
            print("查询所有学生")
        elif num == 3:
            print("查询某一个学生")
        elif num == 4:
            print("修改某一个学生")
        elif num == 5:
            print("删除某一个学生")
        elif num == 6:
            print("退出系统")
            break
        else:
            print("输入错误,请重新输入")
```

```
# 调用主函数
main()
```

4.添加学生信息:

- 数据存储：`[{}, {}, {}]`

  ```
  user_list = [{'name':'mike', 'age':34, 'tel':110},
               {'name':'yoyo', 'age':32, 'tel':120}]
  ```

- 添加学生：
  - `for 字典元素 in user_list:` 再判断用户是否存在列表中，不存在才添加学生名片
  - 列表.append(字典)
- 代码

```
# 0. 函数的外面，定义一个全局变量(列表)，用于保存用户信息
user_list = [{"name": "rose", "age": 20, "tel": "222"},
```

```
                {"name": "mike", "age": 21,
"tel": "111"}]


def add_stu():
    # 1. 输入用户信息：姓名、年龄、电话
    name = input("请输入姓名: ")
    age = int(input("请输入年龄: "))
    tel = input("请输入电话: ")
    # 2. 通过for遍历，取出某个元素后，这个元素就
是字典
    for user_dict in user_list:
        # 2.1 字典['name'] == 用户输入的名字，
是否相等，相等则跳出循环
        if user_dict['name'] == name:
            print("添加的用户已存在,无需添加")
            # 2.2 break跳出循环
            break
    # 3. for中的else 如果用户不存在列表中，添加
用户字典到列表
    else:
        # 3.1 创建字典
        my_dict = {"name": name, "age":
age, "tel": tel}
        # 3.2 追加列表
        user_list.append(my_dict)
```

5.显示所有学生信息:

- 显示所有学生：`for 索引位置，字典 in enumerate(user_list)：`

```
def show_stu():
    # 1. 遍历前，打印一些提示信息：序号     姓名     年龄   电话
    print("序号\t\t姓名\t\t\t年龄\t\t电话")
    # 2. 遍历 for 索引位置，字典 in enumerate(user_list)：
    for i, user_dict in enumerate(user_list):
        # 2.1 打印一个用户的信息 索引位置+1, user_dict[‘name’]……
        print(f"{i + 1}\t\t{user_dict['name']}\t\t{user_dict['age']}\t\t{user_dict['tel']}")
```

6.查询某个学生:

- 查询某个学生
  - `for 字典元素 in user_list:` 找到打印用户信息，然后break跳出循环
  - 没有for的else提示没有此用户

```
# 查询某一个学生
def find_stu():
    # 1. 输入姓名
    f_name = input("请输入需要查询的学生姓名:
")
    # 2. 通过for遍历，取出一个字典user_dict
    for user_dict in user_list:
        # 2.1 user_dict['name']和输入姓名判断
        if user_dict['name'] == f_name:
            # 2.1.1 如果相等，输出用户信息，退
出循环
            print("查询到学生信息: ")
            print(f"
{user_dict['name']}\t\t{user_dict['age']}\t
\t{user_dict['tel']}")
            break
    # 3. for中的else，循环执行完毕，没有break，
说明用户不存在，提示一下
```

```
    else:
        print("用户不存在,请重新输入")
```

7. 修改某个学生

- 修改某个学生
  - `for 索引位置，字典 in enumerate(user_list)：` 找到通过索引位置定位修改，然后break
  - 没有for的else提示没有此用户

```
# 更新某个学生信息，根据输入的姓名匹配哪个学生
def update_stu():
    # 1. 输入需要修改的用户姓名
    name = input("请输入需要修改的学生姓名：")
    # 2. for遍历，带索引的遍历   i, user_dict in user_list
    for i, user_dict in enumerate(user_list):
        # 2.1 如果user_dict['name']和输入用户名字相等
        if user_dict['name'] == name:
            # 2.1.1 重新输入新的姓名、年龄、电话
```

```
                # new_name = input("请输入新的学
生姓名：")
                # new_age = int(input("请输入新
的学生年龄："))
                # new_tel = input("请输入新的学生
电话：")

                # 2.1.2 方法1：对
user_dict['name'] = 新的name ...
                # user_dict['name'] = new_name
                # user_dict['age'] = new_age
                # user_dict['tel'] = new_tel

                # 2.1.2 方法2：对user_list[i]
['name'] = 新的name
                # user_list[i]['name'] =
new_name
                # user_list[i]['age'] = new_age
                # user_list[i]['tel'] = new_tel

                user_list[i]['name'] =
input("请输入新的学生姓名：")
                user_list[i]['age'] =
int(input("请输入新的学生年龄："))
                user_list[i]['tel'] = input("请
输入新的学生电话：")
```

```
            # 2.1.3 ……、修改成功打印、break跳出循环
            print("修改成功")
            break
    # 3. for中的else 输入的用户不在列表
    else:
        print("当前学习不存在, 请重新输入")
```

8. 删除某个学生

- 删除某个学生
  - `for 索引位置, 字典 in enumerate(user_list):` 找到通过索引位置定位删除，然后break
  - 没有for的else提示没有此用户

```
# 删除某个学生，根据输入的姓名
def del_stu():
    # 1. 输入用户姓名
    name = input("请输入学生姓名: ")
    # 2. for遍历, 带索引的遍历   i, user_dict
```

```
    for i, user_dict in
enumerate(user_list):
        # 2.1 如果user_dict['name']和输入用户
名字相等
        if user_dict['name'] == name:
            # 2.1.1 del 列表[i], 同时break跳
出循环
            del user_list[i]
            print("删除成功")
            break
    # 3. for中else 输入的用户在列表中，不存在
    else:
        print("输入的学生姓名不存在,请重新输入")
```