16.6.2 交换变量值

需求：有变量 `a=10` 和 `b=20` ，交换两个变量的值。

- 方法一

借助第三变量存储数据

```
# 1、定义中间变量
c = 0
# 2、将a的数据存储到c
c = a
# 3、将b的数据20复制到a，此时a=20
a = b
# 4、将之前c的数据10赋值到b，此时b = 10
b = c
print(a) # 20
print(b) # 10
```

- 方法二

```
a, b = 1,2
a, b = b,a
print(a) # 2
print(b) # 1
```

16.6.3 引用

16.6.3.1 了解引用

在python中，值是靠引用传递来的。

我们可以用 `id` 来判断两个变量是否为同一个值的引用。我们可以将id值理解为那块内存的地址标识。

```
# 1、int类型
a = 1
b = a
print(b) # 1
print(id(a)) # 1617604832
print(id(b)) # 1617604832

a = 2
print(b) # 1、说明int类型为不可变类型
```

16.6.3.2 可变类型

```python
#2、列表
aa = [10,20]
bb = aa

print(id(aa)) # 1470481080200
print(id(bb)) # 1470481080200

aa.append(30)
print(bb) # [10, 20, 30]  列表为可变类型

print(id(aa)) # 2125878697864
print(id(bb))  #2125878697864
```

16.6.3.3 引用当做实参

```python

def test1(a):
    print(a) # 100
    print(id(a)) # 1617608000

    a += a

    print(a) # 200
    print(id(a)) # s

# int:计算前后id值不同
b = 100
test1(b)

#列表：计算前后id值不同
c = [11,12]
test1(c)

"""
[11, 12]
2728354815880
[11, 12, 11, 12]
2728354815880

"""
```

16.6.3.4 可变和不可变类型

所谓可变类型与不可变类型是指：数据能够直接进行修改，如果能直接修改那么就是可变，否则是不可变。

- 可变类型

  列表

  字典

  集合

- 不可变类型

整型

浮点型

字符串

元组

十七、函数加强

目标

- 应用：学员管理系统
- 递归
- lambda表达式
- 高阶函数

17.1 应用：学员管理系统

1.1 吸引简介

需求：进入系统显示功能界面，功能如下：

- 添加学员
- 删除学员
- 修改学员信息
- 查询学员信息

系统共6个功能，用户根据自己需求选取

17.2 步骤分析

1、显示功能界面

2、用户输入功能序号

3、根据用户输入的功能序号，执行不能的功能（函数）

3.1 定义函数

3.2 调用函数

17.3 需求实现

17.3.1 显示功能界面

定义函数 `print_info` ，负责显示系统功能。

```
# 定义功能界面函数
def info_print():
    print('请选择功能-------')
    print('1、添加学员')
    print('2、删除学员')
    print('3、修改学员')
```

```
    print('4、查询学员')
    print('5、显示所有学员')
    print('6、退出系统')
    print('-' * 20)


# 系统功能需要循环使用，直到用户输入6，退出系统
while True:
    # 1、显示功能界面
    info_print()

    # 2、用户输入功能序号
    user_num = int(input('请输入功能序号：'))

    # 3、按照用户输入的功能序号，执行不同的界面
    # 如果用户输入1、执行添加..... ----多重判断
    if user_num == 1:
        print('添加')
    elif user_num == 2:
        print('删除')
    elif user_num == 3:
        print('修改')
    elif user_num == 4:
        print('查询')
    elif user_num == 5:
        print('显示所有')
    elif user_num == 6:
        print('退出系统')
        break
    else:
        print('输入的功能序号有误')
```

17.3.2 定义不同功能的函数

所有功能函数都是操作学员信息，所有存储所有学员信息应该是一个全局变量，数据类型为列表。

```
info = []
```

17.3.2.1 添加学员

- 需求分析

  1、接收用户输入学员信息，并保存

  2、判断是否添加学员信息

  2.1 如果学员姓名已经存在，则报错提示

  2.2 如果学员姓名不存在，则准备空字典，将用户输入的数据追加的字典，再列表追加字典数据

  3、对应的if条件成立的位置调用该函数

- 代码实现

```
# 等待存储所有学员的信息
```

```python
info = []
# 添加学员信息的函数
def add_info():
    """添加学员函数"""
    # 添加用户输入学员信息
    # 1 用户书臣：学号、姓名、手机号

    new_id = input('请输入学号：')
    new_name = input('请输入姓名：')
    new_tel = input('请输入手机号：')

    # 声明info是全局变量
    global info
    #2 判断是否添加这个学员：如果学员姓名一斤存在报错提示：如果不存在则添加数据
    # 2.1 不允许姓名重复：判断用户输入的姓名和 列表里面字典的name对应的值 相等 提示
    for i in info:
        if new_name == i['name']:
            print('此用户已经存在')
            return # 此时，退出当前函数，后面添加信息的代码不执行

    # 2.2 如果输入的姓名不存在，添加数据：准备空字典，字典新增数据，列表追加字典
    info_dict = {}

    # 字典新增数据
    info_dict['id'] = new_id
    info_dict['name'] = new_name
    info_dict['tel'] = new_tel
    print(info_dict) # 打印一下

    # 列表追加字典
    info.append(info_dict)
    print(info)
```

17.3.2.2 删除学员

- 需求分析

按用户输入的姓名进行删除

1、用户输入目标学员姓名

2、检查这个学员是否存在

2.1 如果存在，则列表删除这个数据

2.2 如果不存在，则提示“该用户不存在”

3、对应的if条件成立的位置调用该函数

- 代码实现

```python
# 删除学员信息的函数
def del_info():
```

```
    """删除学员信息"""
    # 1、用户输入目标学员姓名
    del_name = input('请输入要删除的学员姓名：')

    global info
    # 2、检查这个学员是否存在
    for i in info:
        if del_name == i['name']:
            info.remove(i)
            break
    else:
        print('该学员不存在')
    print(info)
```

17.3.2.3 修改学员

- 需求分析

1、用户输入目标学员姓名

2、检查这个学员是否存在

2.1 如果存在，则修改这个学员的信息，例如手机号

2.2 如果不存在，则报错

3、对应的if条件成立的位置调用该函数

```
# 修改学员信息的函数
def modify_info():
    """修改函数"""
    # 1、用户输入目标学员姓名
    modify_name = input('请输入要修改学员的姓名：')
    global info
    # 2、检查这个学员是否存在
    #    2.1 如果存在，则修改这个学员的信息，例如手机号
    #    2.2 如果不存在，则报错
    for i in info:
        if modify_name== i['name']:
            # 将tel这个key修改值，并终止此循环
            i['tel'] = input('请输入新的手机号：')
            break
    else: # 循环都没执行，自然不存在
        print('学员不存在')
    print(info)
```

17.3.2.4 查询学员信息

- 需求分析

1、用户输入目标学员名称

2、检查学员是否存在

```
    2.1 如果存在，则显示这个学员的信息
```

2.2 如果不存在，则报错

3、对应的if条件成立的位置调用该函数

- 代码实现

```
def search_info():
    """查询学员信息"""
    # 1、用户输入目标学员名称
    search_name = input('请输入要查找的学员姓名：')
    global  info
    # 2、检查学员是否存在
    #    2.1 如果存在，则显示这个学员的信息
    #    2.2 如果不存在，则报错
    for i in info:
        if search_name == i['name']:
            print('查到找学习信息如下：--------')
            print(f"该学员的学号为{i['id']},姓名是{i['name']}，手机号是{i['tel']}")
            break
    else:
        print('不存在')
    # 3、对应的if条件成立的位置调用该函数
```

17.3.2.5 显示所有成员信息

- 需求分析

打印所以学员信息

- 代码实现

```
# 显示所有学员信息
def print_all():
    """显示所有学员信息"""
    #1、打印提示字
    print('学号\t姓名\t手机号')
    #2、打印所有学员的数据
    for i in info:
        print(f"{i['id']}\t {i['name']}\t {i['tel']}")
```

17.3.2.6 退出系统功能

在用户输入序号6的时候要退出系统，代码如下：

```
elif user_num == 6:
    exit_fla = input('确定要退出系统 yes or no:')
    if exit_fla == 'yes':
        break
```

十八、递归

18.1 递归的应用场景

递归是一种编程思想，应用场景：

1. 在我们日常开发中，如果要遍历一个文件夹下面所有的文件，通常会使用递归来实现；
2. 在后续的算法学习中，很多算法都离不开递归，例如：快速排序。

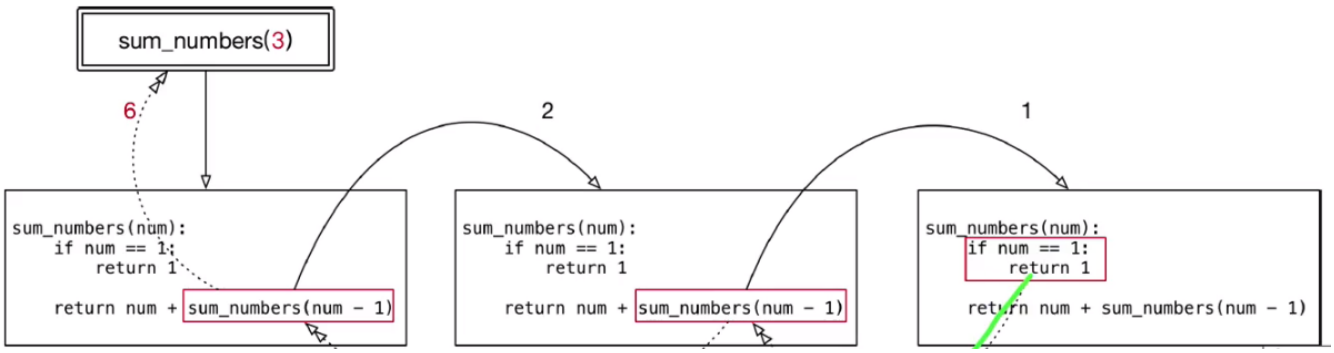
18.2 递归的特点

- 函数内部自己调用自己
- 必须有出口

18.3 应用：3以内数字累加和

虽然方法很多（sun(range(4)),while等实现），但是您能想到递归的方法嘛？？？滋滋

```python
#3+2+1
def sum_numbers(num):
    # 1、如果是1，直返返回1 -- 出口
    if num ==1:
        return  1
    # 2、如果不是1， 重复执行累加并返回结果
    return num + sum_numbers(num-1)
sum_result = sum_numbers(3)
print(sum_result)
```



十九、lambda表达式

19.1 lambda的应用场景

如果一个函数有一个返回值，并且只有一句代码，可以使用lambda简化

19.2 lambda语法

```
lambda 参数列表 ： 表达式
```

注意：

1. lambda表达式的参数可有可无，函数的参数在 lambda表达式中完全适用。
2. lambda表达式能接收任何数量的参数但只能返回一个表达式的值。

快速入门：

```
#需求：函数 返回值100
def fn1():
    return 200
result = fn1()
print(result)


# lambda 匿名函数
# lambda 参数列表 ：表达式
fn2 = lambda : 100
print(fn2) # lambda内存地址
#100返回值 调用函数
print(fn2())
```

注意：直接打印 lambda表达式，输出的是此 lambda的内存地址

19.3 实例

19.3.1 计算a+b

```
def add(a,b):
    return a+b
result = add(1,2)
print(result)

```

不难发现代码太多了。

19.3.2 lambda实现

```
fn1 = lambda a,b : a+b
print(fn1(1,2))
```

19.4 lambda参数

19.4.1 无参数

```python
fn1 = lambda : 100
print(fn1()) # 100
```

19.4.2 一个参数

```python
fn1 = lambda a:a
print(fn1('hello word'))
```

19.4.3 默认参数

```python
fn1 = lambda a,b,c=100:a+b+c
print(fn1(10,20)) # 100
```

19.4.4 可变参数：*args

参数个数不定，根据程序员输入界定

```python
fn1 = lambda *args: args
print(fn1(10,20,30))
```

注意：这里可变参数传入到lambda之后，返回值为元组

19.4.5 可变参数：**kwargs

```python
fn1 = lambda **kwargs :kwargs
print(fn1(name='python',arg=20))
```

注意：这里可变参数传入到lambda之后，返回值为字典

19.5 lambda应用

19.5.1 带判断的lambda

```python
fn1 = lambda a, b: a if a>b else b
print(fn1(1,2))
```

19.5.2 列表数据按字典key的值排序

```python
students = [
    {'name':'jjk', 'age':20},
    {'name':'ROSE', 'age':18},
```

```
    {'name':'Jack', 'age':23}
]

#按name值升序排列
students.sort(key=lambda x:x['name'])
print(students)

#按name值升序排列
students.sort(key=lambda x : x['name'],reverse=True)
print(students)

# 按age值升序排序
students.sort(key=lambda x : x['age'])
print(students)
```

二十、高阶函数