Day05 课堂笔记

课程之前

复习:

根据下面的笔记内容快速回顾之前所学知识。老师课堂上就不讲了

```
for x in 元组/列表:
    print(x)

字典的键
for k in 字典.keys():
    print(k)

字典的值
for v in 字典.values():
    print(v)

字典的键值对
for k, v in 字典.items():
    print(k, v)  # k 键,  v 值

while 循环 可以遍历列表 元组
i = 0  # 表示下标

while i < len(元组/列表):
    print(列表[i])
    i += 1

def say_hello():  # 定义
    print('hello')
    print('hello')
    print('hello')


say_hello()  # 调用


返回值:  函数中得出的数据，想要在函数外边用，需要使用返回值返回。

想要使用 模块或者包中的内容，所以要导入。
import 模块名
模块名.功能名

from 模块 import 功能
from 包名.模块名 import 功能
```

```
def 函数名(参数, ...):
    函数中的代码
    return xxx  # 看 return 之后的数据，是否还要使用，如果使用就使用 return
```

作业

```
range(n)  ---> [0, n)
range(start, end, step)     ---> [start, end) 步长是 step,默认 1

range(1, 5, 2)  # [1, 3]
range(1, 101)   # 1-100
```

今日内容

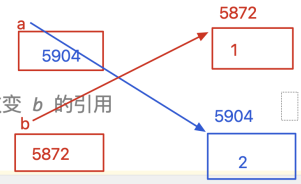
```
变量的进阶
-   引用 -- 变量 = 数据值 本质(了解)
-   可变类型与不可变类型(记住)
-   局部变量和全局变量(理解掌握特点)
函数的进阶
-   函数返回多个数据
-   默认参数,缺省参数(形参)
-   不定长参数,可变参数,多值参数(形参)
-   匿名函数(lambda 关键字)
案例：学生管理系统
```

变量的进阶

引用[了解]

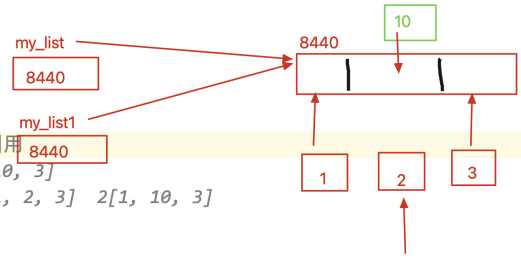
```
1，定义变量的时候，变量和数据 都会在内存开辟空间
2，变量所对应的内存空间中存储的是 数据所在内存的地址 (平时理解为 将数据存储到变量中即可)
3. 变量中保存数据的地址,就称为是引用
4，Python 中所有数据的传递,传递的都是引用(即地址)
5，赋值运算符(=)，会改变变量的引用，即只有 = 可以修改变量的引用
6，可以使用 id(变量)  函数,查看变量的引用
```

```python
a = 1  # 本质：将数据 1 的地址保存到变量 a 中，通常理解：将 1 保存到 a
b = a  # 本质：将 变量 a中的引用保存到变量 b中.  通常理解 将 a 的值给 b
print(f"a: {a}, {id(a)}")
print(f"b: {b}, {id(b)}")
a = 2   # 本质：将数据 2 的地址保存到变量a中，只是改变 a的引用,即改变 a的值，没有改变 b 的引用
print(f"a: {a}, {id(a)}")  # 2
print(f"b: {b}, {id(b)}")  # 1
```




```
Run: hm_01_引用
/Users/nl/opt/anaconda3/envs/py36/bin/python /Users/nl/Desktop/20211115_gz_py/day05_函数/04-代码/hm_01_引用.py
a: 1, 4391875872
b: 1, 4391875872
a: 2, 4391875904
b: 1, 4391875872
```

```python
my_list = [1, 2, 3]
my_list1 = my_list
print(f'my_list :{my_list}, {id(my_list)}')
print(f'my_list1:{my_list1}, {id(my_list1)}')
my_list[1] = 10  # 修改 列表my_list 中下标为 1 位置的引用
print(f'my_list :{my_list}, {id(my_list)}')   # [1, 10, 3]
print(f'my_list1:{my_list1}, {id(my_list1)}')  # 1 [1, 2, 3]  2[1, 10, 3]
```




```
Run: hm_01_引用
b: 1, 4377826592
a: 2, 4377826624
b: 1, 4377826592
my_list :[1, 2, 3], 140212509058440
my_list1:[1, 2, 3], 140212509058440
my_list :[1, 10, 3], 140212509058440
my_list1:[1, 10, 3], 140212509058440
```

可变类型与不可变类型[记住]

根据内存中的数据是否允许修改,将数据类型分为可变类型与不可变类型
简单理解：不使用等号,能不能修改数据值
可变类型：可以修改
- 列表(list) list.append()
- 字典(dict) dict.pop()
- 集合(set)

不可变类型：不允许修改
- 数字类型(int , float , bool)
- 字符串(str)
- 元组(tuple)

```python
my_tuple = (1, 2, [10, 20])  # 元组中 存储的 1 的地址, 2 的地址, 列表的地址

print(my_tuple, id(my_tuple), id(my_tuple[-1]), id(my_tuple[-1][-1]))
my_tuple[-1][-1] = 30  # 修改的列表中最后一个位置的引用
print(my_tuple, id(my_tuple), id(my_tuple[-1]), id(my_tuple[-1][-1]))
```

局部变量和全局变量

根据变量定义的位置将变量分为局部变量和全局变量

局部变量

1，在函数内部定义的变量,称为是局部变量
2，特点
- 2.1 局部变量,只能在当前函数内部使用
- 2.2 可以在不同函数内定义名字相同的局部变量
3，生命周期(使用范围)
- 3.1 在函数执行(调用)的时候被创建
- 3.2 函数执行结束被销毁(删除)
4，形参可以认为是局部变量
5，如果想要在函数外部使用局部变量的值，使用 `return` 返回

```python
def func1():
    num = 10  # 局部变量
    print(num)


def func2():
    num = 20
    print(num)


if __name__ == '__main__':
    func1()  # 10
    func2()  # 20
    func1()  # 10
```

全局变量

1，在函数外部定义的变量
2，特点
- 2.1 全局变量 可以在任意函数内访问(读取)
- 2.2 想要在函数内部修改全局变量的引用,需要使用 `global` 关键字声明(使用 `global` 关键字可以声明为全局变量)
- 2.3 如果在函数内部出现和全局变量名字相同的局部变量,在函数内部使用的是局部变量
3，生命周期
- 代码执行的时候 创建，执行结束销毁

```python
# 定义全局变量
g_num = 10


def func_1():
    print(g_num)  # 使用全局变量
```

```python
def func_2():
    g_num = 20  # 定义局部变量
    print(g_num)


def func_3():
    global g_num   # 声明为全局变量
    g_num = 30
    print(g_num)


if __name__ == '__main__':
    print(g_num)  # 10
    func_1()  # 10
    func_2()  # 20
    func_1()  # 10
    print(g_num)  # 10
    func_3()  # 30  修改了全局变量，将全局变量的值改为30 了
    func_1()  # 30
    g_num = 100
    func_1()  # 100  修改全局变量的值
    func_2()  # 20  局部变量
    func_3()  # 30
    func_1()  # 30
```

```python
def func1():
    list1.append(10)


def func2():
    list1 = [1, 1]  # 定义局部变量，不影响全局变量
    list1.append(0)


def func3():
    global list1  # 全局变量
    list1.pop()  # 删除最后一个数据


def func_5():
    list1.pop()  # 用的全局变量,没有改引用


def func4():
    global list1  # 全局变量
    list1 = [1]


if __name__ == '__main__':

    list1 = [1, 2]
```

```
    func1()
    print(list1)  # ①[1, 2]  ②[1, 2, 10](√)  ③报错
    func2()
    print(list1)   # ① [1, 1, 0] ②[1, 2, 10](√) ③报错
    func3()
    print(list1)  # [1, 2]
    # func_5()
    # print(list1)  # ②[1, 2]   ①[1]对
    func4()
    print(list1)  # [1]
```

函数进阶

函数返回多个数据值

return 关键字两个作用:
1. 返回数据值
2. 结束函数的运行
函数中如果想要返回多个数据值，一般是组成元组进行返回

```
def calc(a, b):
    """返回两个数的和及差"""
    return a + b, a - b


result = calc(10, 20)
print(result)
```

函数传参的方式

是指如何将 实参的值传递给形参

位置传参

在函数调用的时候按照 形参的顺序将实参的值传递给形参

关键字传参

在函数调用的时候，指定将实参传递给哪个形参

混合使用

1. 位置参数必须写在关键字参数的前边，关键字参数必须放在位置参数的后边
2. 同一个形参只能接收一个实参值(不能即使用位置传参和关键字传参给同一个形参传参)

```
def show_info(name, age):
    print(f"name:{name}, age: {age}")


# 位置传参
show_info('小明', 18)

# 关键字传参
show_info(age=18, name='张三')

# 混合使用
show_info('李四', age=17)
```

缺省参数(默认参数)

- 定义

  在函数定义的时候，给形参一个默认的数据值,这个参数就是缺省参数(默认参数)

- 特点

  在函数调用的时候，缺省参数可以不用传递实参值.
  1. 如果不传实参值，使用的就是默认值
  2. 如果传递实参值，使用的就是传递的实参值

- 注意点

  缺省参数必须写在 普通参数的后边

- 代码

```
"""
定义 show_info 参数 姓名,年龄，性别，将年龄设置为默认参数 18，性别设置为默认 保密
"""


def show_info(name, age=18, sex='保密'):
    print(name, age, sex)


# 调用

show_info('张三', 18, '男')
```

```
# 李四
show_info('李四')
# 王五 19
show_info('王五', 19)
# 赵六  男
show_info('赵六', sex='男')
```

多值参数(可变参数/不定长参数)

1，在函数定义的时候,不确定在调用的时候,实参有多少个,此时可以使用 多值参数
2，在普通的参数前边加上一个 *，这个参数就变为 多值参数
3，这个参数可以接收任意多个位置传参的数据，类型 元组
4，这个形参一般写作 args(arguments)，即 *args

```
print(1)
print(1, 2)
print(1, 2, 3)
print(1, 2, 3, 4)
```

- 参数顺序

```
# 普通，缺省，多值
def 函数名(普通，*args，缺省):
    pass
```

- 代码

```
def func(*args):
    print(args)


func()   # ()
func(1, 2, 3)  # (1, 2, 3)
```

- 练习

定义一个函数,my_sum,作用,可以对任意多个数字进行求和计算.

```
def my_sum(*args):   # args 是元组
    num = 0
    for i in args:
        num += i

    return num


print(my_sum())  #
```

```python
print(my_sum(1))  # 1
print(my_sum(1, 2))  # 3
print(my_sum(1, 2, 3))  # 6

my_tuple = (1, 2, 3, 4, 5)
# 需求对 元组中的所有数据使用 my_sum 进行求和
# 想要把列表(元组) 中的数据作为位置参数进行传递，只需要在列表(元组)前边加上一个 * ,进行拆包即可
print(my_sum(*my_tuple))  # 15
```

匿名函数

匿名函数：使用 lambda 关键字 定义的表达式,称为匿名函数.

- 语法

```python
lambda 参数, 参数: 一行代码  # 只能实现简单的功能,只能写一行代码
# 匿名函数 一般不直接调用，作为函数的参数使用的
```

- 代码

```python
"""定义匿名函数,可以对两个数字进行求和"""

func = lambda a, b: a + b  # 将匿名函数 地址保存到变量 func 中，所以 func 就指代函数

print(func(1, 2))
```

PEP 8: do not assign a lambda expression, use a def

Reformat file ⌥⇧⏎ More actions... ⌥⏎

不要给 lambda 表达式赋值，使用def

应用： 作为函数的参数，这个参数书写一次， 因为使用def定义太麻烦

- 练习

```python
# 1，定义匿名函数，参数为两个整数数字，求两个数字的乘积
lambda a, b: a * b
# 2，定义匿名函数，参数为一个字典，返回字典中 键为 age 的值
lambda x: x.get('age')
lambda x: x['age']
```

- 匿名函数的应用

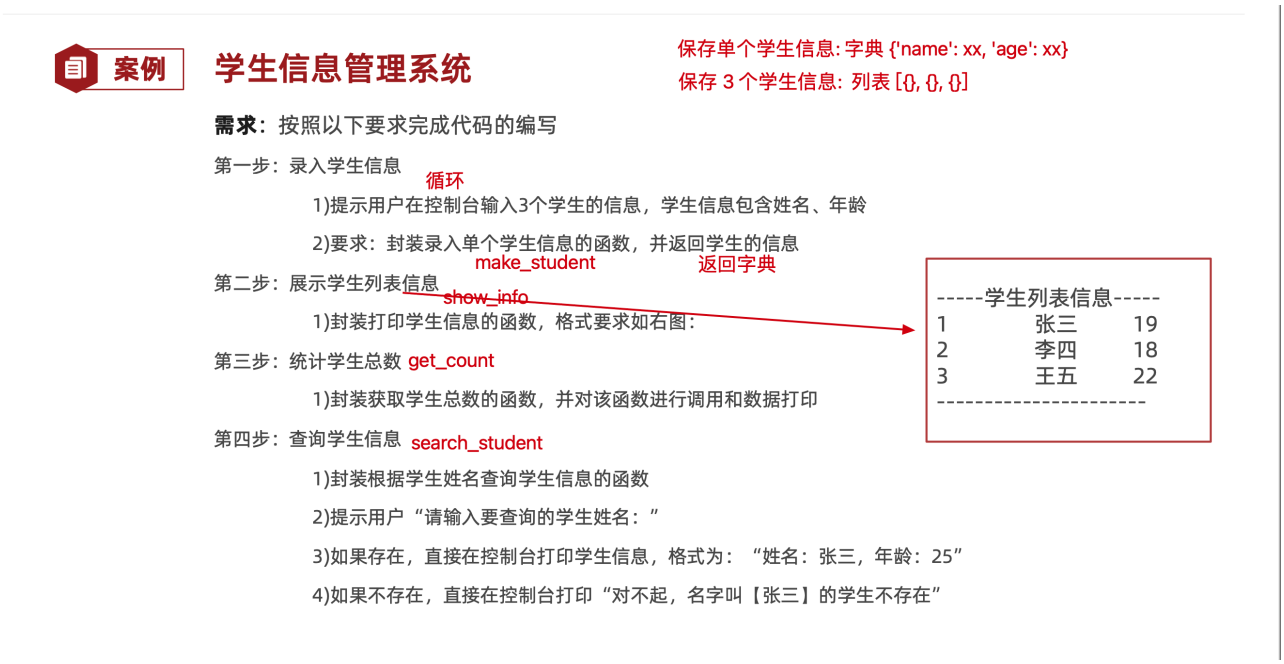
```python
user_list = [
    {'name': '张三', 'age': 22, 'title': '测试工程师'},
    {'name': '李四', 'age': 24, 'title': '开发工程师'},
    {'name': '王五', 'age': 21, 'title': '测试工程师'}
]


# user_list.sort()  # 只能对数字,字符串排序
# 根据字典的 age 键 排序
# 想要对列表中的字典排序,需要 key 形参来指定根据字典中的什么键排序
# key 这个参数需要传递一个函数,使用匿名函数
# 列表.sort(key=lambda x: x['键'])
```

```python
def func(x):
    return x['age']


user_list.sort(key=lambda x: x['age'])
# user_list.sort(key=func)
print(user_list)
```

案例




```python
# 定义一个列表，保存所有学生信息
# stu_list = []
stu_list = [{'name': 'aa', 'age': '11'}, {'name': 'bb', 'age': '22'}, {'name': 'cc', 'age':
'33'}]


def make_student():
    """录入单个学生信息"""
    name = input('请输入姓名:')
    age = input('请输入年龄:')
    # 将学生信息存入字典
    stu_dict = {"name": name, "age": age}
    # 返回单个学生信息
    return stu_dict


def show_stu_info():
    """展示学生信息"""
    print('---------学生列表信息-----------')
    j = 1  # 初始序号
    for stu_dict in stu_list:  # stu_dict 字典
        print(f"{j}\t\t{stu_dict.get('name')}\t\t{stu_dict.get('age')}")
```

```
        j += 1  # 修改序号
    print('------------------------------')


def get_student_counts():
    """获取学生的数量"""
    return len(stu_list)


def search_student():
    """查询学生的信息"""
    name = input('请输入要查询的学生姓名:')
    for stu_dict in stu_list:
        if name == stu_dict.get('name'):
            # 找到了这个学生
            print(f'姓名:{name}, 年龄: {stu_dict.get("age")}')
            # 终止
            return  # 结束函数的执行

    # 写在循环的外边
    print(f'对不起, 名字叫 [{name}]的学生不存在')


if __name__ == '__main__':
    # 录入三个学生信息
    # for i in range(3):
    #     stu = make_student()
    #     # 需要将单个学生添加到列表
    #     stu_list.append(stu)
    #
    # print(stu_list)
    # 展示学生信息
    show_stu_info()
    # 获取学生数量
    print('学生总数为: ', get_student_counts())
    search_student()
```