12.3 max()和min()

```
# 1、字符串
str1 = 'abdfdf'
# 2、列表
list1 = [10,20,30,40,50]
print(max(str1))
print(max(list1))

print(min(str1))
print(min(list1))
```

12.4 range(start,end,step)

```
for i in range(1,10,1):
    print(i) # 1-9
# print(range(1,10,1)) # 返回的可迭代的对象
for i in range(1,10,2):
    print(i) # 1 3 5 7 9

for i in range(10):
    print(i) # 1-9
# 1、如果不写开始，默认从0开始
# 2、如果不写步长，默认为1
```

注意：range()生成的序列不包含end数字。

12.5 enumerate()

- 语法

```
enumerate(可遍历对象, start=0)
```

注意：start参数用来设置遍历数据的下标的起始值，默认为0

- 快速体验

```
list1 = ['a', 'b', 'c', 'd', 'e']
for i in enumerate(list1):
    print(i)
for index, char in enumerate(list1,start=1):
    print(f'下标是{index},对应的字符是{char}')

"""
(0, 'a')
(1, 'b')
(2, 'c')
```

```
(3, 'd')
(4, 'e')
"""

"""
下标是1,对应的字符是a
下标是2,对应的字符是b
下标是3,对应的字符是c
下标是4,对应的字符是d
下标是5,对应的字符是e

"""
```

说明：enumerate 返回的结果是元组，元组第一个数据是原迭代对象的数据对应的下标，元组第二个数据是元迭代对象的数据。

十三、容器类型的转换

13.1 tuple()

作用：将某个序列转换成元组

```
list1 = [10,20,30,40,50]
s2 = {100,200,300,400,50}

print(tuple(list1))
print(tuple(s1))
```

13.2 list()

作用：将某个序列转换成列表

```
t1 = ('a', 'b', 'c', 'd', 'e')
s2 = {100,200,300,400,50}

print(list(t1))
print(list(s1))

```

13.3 set()

作用：将某个序列转换成集合

```python
list1 = [10,20,30,40,50]
t1 = ('a', 'b', 'c', 'd', 'e')
print(set(list1))
print(set(t1))
```

十四、推导式

目标：

- 列表推导式
- 字典推导式
- 集合推导式

14.1 列表推导式

作用：用一个表达式创建一个有规律的列表或控制一个有规律列表。

列表推导式又叫列表生成式。

快速体验：

while实现

需求：创建一个0-10的列表

```python
#1、准备一个空列表
list1 = []
#2、书写循环，一次追加数字到空列表list1中
i = 0
while i<10:
    list1.append(i)
    i += 1
print(list1)
```

for循环实现

```python
list1 = []
for i in  range(10):
    list1.append(i)
print(list1)
```

列表推导式实现

```python
list1 = [i for i in range(10)]
print(list1)
```

带if的列表推导式

需求：创建0-10的偶数列表

- 方法一：range()步长实现

```
list1 = [i for i in range(0,10,2)]
print(list1)
```

- 方法二：if实现

```
list1 = []
for i in range(10):
    if i%2 ==0:
        list1.append(i)

list1 = [i for i in range(10) if i % 2 ==0]
print(list1)
```

多个for循环实现列表推导式

```
# 需求：[(1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]

list1 = []
for i in range(1,3):
    for j in range(3):
        # 列表里面追加元组，循环前面准备一个空列表，然后这里追加元组数据到列表
        list1.append((i,j))
print(list1)

list1 = [(i,j) for i in range(1,3) for j in range(3)]

# 多for的列表推导式等同于for循环嵌套
```

14.2 字典推导式

思考：如果有如下两个列表：

```
list1 = ['name', 'age', 'gender']
list2 = ['Tom', 23, 'man']
```

如何快速合并成一个字典？

答：字典推导式

字典推导式作用：快速合并列表为字典或提取字典中目标数据。

快速体验

1、创建一个字典：字典key是1-5数字，value是这个数字的2次方。

```
dict1 = {i : i**2 for i in range(1,5)}
print(dict1)
"""
{1: 1, 2: 4, 3: 9, 4: 16}
"""
```

2、将两个列表合并成一个字典

```
list1 = ['name', 'age', 'gender']
list2 = ['Tom', 23, 'man']

dict1 = {list1[i]:list2[i] for i in range(len(list1))}
print(dict1)

"""
{'name': 'Tom', 'age': 23, 'gender': 'man'}
"""
# 总结:
# 如果两个列表数据个数相同，len统计任何一个列表的长度都可以
# 如果两个列表数据个数不同，len统计数据多的列表数据个数会报错；len统计数据少的列表数据个数不会报错
```

3、提取字典中目标数据

```
counts = {'MBP':268, 'hp':235, 'DELL':343, 'Lenovo':123,'acer':99}
# 需求：提取上述电脑数量大于200的字典数据
count1 = {key:value for key,value in counts.items() if value>=200}
print(count1)

# {'MBP': 268, 'hp': 235, 'DELL': 343}
```

14.3 集合推导式

需求：创建一个集合，数据为下标列表的2次方

```
list1 = [1,1,2]
```

代码如下：

```
list1 = [1,1,2]
set1 = {i**2 for i in list1}
print(set1)
#{1, 4}
```

注意：集合有数据去重功能

总结

```
#列表推导式
[xx for xx in range()]

# 字典推导式
{xx1:xx2 for ... in ...} # 注意字典是键值对存在

# 集合推导式
{xx for xx in ...}
```

十五、函数一

目标：

- 函数的作用
- 函数的使用步骤
- 函数的参数作用
- 函数的返回值作用
- 函数的说明文档
- 函数嵌套

15.1 函数的作用

需求：用户到ATM机取钱：

1. 输入密码后显示"选择功能"界面
2. 查询余额后显示"选择功能"界面
3. 取2000钱后显示"选择功能"界面

特点：显示选择功能”界面需要重复输出给用户，怎么实现?

函数就是将一段具有独立功能的代码块整合到一个整体并命名，在需要的位置调用这个名称即可完成对 应的需求。

函数在开发过程中，可以更高效的实现代码重用。

15.2 函数的使用步骤

15.2.1 定义函数

```
def 函数名称(参数):
    代码1
    代码2
    ...
```

15.2.2 调用函数

```
函数名(参数)
```

> 注意:

1、不同的需求，参数可有可无

2、在python中，函数必须先定义后调用

15.2.3 快速体验

需求：复现ATM取钱功能

1、搭建整体框架（复现需求）

```
print('密码正确登录成功')
# 显示“选择功能”界面
print('查询余额完毕')

# 显示“选择功能”界面
print('取了2000大洋')
# 显示“选择功能”界面
```

2、确定“选择功能”界面内容

```
print('余额查询')
print('存款')
print('取款')
```

3、封装“选择功能”

注意：一定是先定义函数，后调用函数

```
# 封装ATM机功能选择---定义函数
def select_func():
    print('---请选择功能---')
    print('查询余额')
    print('存款')
    print('取款')
    print()
    print('---请选择功能---')
```

4、调用函数

在需要显示“选择功能”函数的位置调用函数

```
select_func()
```

15.3 函数的注意事项

```
1、先定义函数
2、后调用函数
3、函数的执行流程***
    当调用函数的时候，解释器回到定义函数的地方执行下方缩进的代码，当这些代码执行完，回到调用函数的地方继续向下执行；
    定义函数的时候，函数体内部缩进的代码并没有执行
```

15.4 函数的参数作用

思考：完成需求如下：一个函数完成两个数1和2的加法运算，如何书写程序?

```
def add_num1():
    result = 1+2
    print(result)
# 调用函数
add_num1()
```

思考：上述add_num1函数只能完成数字和2的加法运算，如果想要这个函数变得更灵活，可以计算任 何用户指定的两个数字的和，如何书写程序?

分析：用户要在调用函数的时候指定具体数字，那么在定义函数的时候就需要接收用户指定的数字。函 数调用时候指定的数字和定义函数时候接收的数字即是函数的参数。

```
#定义函数时同时定了接收用户数据的参数num1，num2 被称为形参
def add_num2(num1,num2):
    result = num1 + num2
    print(result)
# 调用函数时传入了真实的数据10和20，真实数据为实参
add_num2(10,20) # 30
```

15.5 函数返回值

例如：我们去超市购物，比如买烟，给钱之后，是不是售货员会返回给我们烟这个商品，在函数中，如 果需要返回结果给用户需要使用函数返回值。

```
def buy():
    return '烟'
# 使用变量保存函数返回值
goods = buy()
print(goods)
# reutrn返回结果给函数调用的位置
# return作用:
# 1、负责函数返回值
# 2、退出当前函数，导致return下方的所有代码（函数体内部）不执行
```

应用

需求：制作一个计算器，计算任意两个字数之和。

```
def sum_num(a,b):
    return a+b
#用result变量保存函数返回值
result = sum_num(10,20)
print(result)
```

15.6 函数的说明文档

思考：定义一个函数后，程序员如何书写程序能够快速提示这个函数的作用？

答：注释 思考：如果代码多，我们是不是需要在很多代码中找到这个函数定义的位置才能看到注释？如果想更方 便的查看函数的作用怎么办？

答：函数的说明文档

注意：函数的说明文档也叫函数的文档说明。

15.6.1 语法

- 定义函数的说明文档

```
def 函数名(参数):
    """说明文档的位置"""
    代码
    ....
```

- 查看函数的说明文档

```
help(函数名)
```

15.6.2 快速体验

```
def sum_num(a,b):
    """求和函数"""
    return a+b

help(sum_num)

# 函数说明文档的高级使用
def sum_num1(a, b):
    """
    求和函数sum_num1
    :param a: 参数1
    :param b: 参数2
    :return:  返回值
    """
    return a+b
```

```
help(sum_num1)
```

15.7 函数的嵌套调用

所谓函数嵌套调用指的是一个函数里面又调用了另外一个函数。

```

def testB():
    print('-------testb start------')
    print('这里是testb函数执行的代码')
    print('--------testb end-------')

def testA():
    print('------testa start--------')
    testB()
    print('------testa end----------')
testA()
```

15.7.1 函数嵌套调用之应用一

1、打印一条横线

```
def print_line():
    print('-'*20)
```

2、打印多条横线

```
def print_line():
    print('-'*20)

def print_lines(num):
    i = 0
    while i<=num:
        print_line()
        i += 1
print_lines(5)
```

15.7.2 函数嵌套调用之应用二

1、求三个函数之和

```
def sum_num(a,b,c):
    return a+b+c
result = sum_num(1,2,3)
print(result)

```

2、求三个数的平均值

```python
def sum_num(a,b,c):
    return a+b+c
def average_num(a,b,c):
    sumResult = sum_num(a,b,c)
    return sumResult /3
averageResult = average_num(1,2,3)
print(averageResult) # 2.0
```

十六、函数二

目标：

- 变量作用域
- 多函数程序执行流程
- 函数的返回值
- 函数的参数
- 拆包和交换两个变量的值
- 引用
- 可变和不可变类型

16.1 变量作用域

变量作用域指的是变量生效的范围，主要分为两类：局部变量和全局变量。

- **局部变量**

所谓局部变量是定义在函数体内部的变量，即只在函数体内部生效。

```python
def testA():
    a = 100
    print(a)
testA() # 100
print(a) # 报错
```

变量a是定义在 testA函数内部的变量，在函数外部访问则立即报错。

局部变量的作用：在函数体内部，临时保存数据，即当函数调用完成后，则销毁局部变量。

- **全局变量**

所谓全局变量，指的是在函数体内、外都能生效的变量。

思考：如果有一个数据，在函数A和函数B中都要使用，该怎么办？

答：将这个数据存储在一个全局变量里面。

```python
# 定义全局变量
a = 100
def testA():
    print(a) # 访问全局变量

def testB():
    print(a) # 访问全局变量
testA()
testB()
```

思考：testB函数需求需改变量a的值为200，如何修改程序?

```python
# 定义全局变量
a = 100
def testA():
    print(a) # 访问全局变量

def testB():
    a = 200
    print(a) # 访问全局变量
testA()
testB()
print(f'全局变量a={a}') # 100
```

思考：在 testB函数内部的a=200中的变量a是在修改全局变量a吗?

答：不是。观察上述代码发现，11行得到a的数据是100，仍然是定义全局变量a时候的值，而没有返回testB函数内部的200。综上：testB函数内部的a=200是定义了一个局部变量。

```python
# 定义全局变量
a = 100
def testA():
    print(a)  # 访问全局变量

def testB():
    # a =200 # 如果直接修改a=200,此时的a是全局还是局部a? ---局部
    # 因为在全局变量（b函数调用后）打印a得到的不是200而是100
    # 想要全局变量a 值是200
    global a # 声明全局变量
    a = 200
    print(a)  # 访问全局变量

testA()
testB()
print(a) # 200
```

16.2 多函数程序执行流程

一般在实际开发过程中，一个程序往往由多个函数（后面知识中会讲解类）组成，并且多个函数共享某 些数据，如下所示：

- 共用全局变量

```
#1、定义全局吧今后
glo_num = 0

def test1():
    global glo_num
    # 修改全局变量
    glo_num = 100

def test2():
    #调用test1函数中修改后的全局变量
    print(glo_num)
#2、调用test1函数，执行函数内部代码：声明和修改全局变量
test1()
#3、调用test2函数，执行函数内部代码：打印
test2() # 100
```

16.3 返回值作为参数传递

- 返回值作为参数传递

```
def test1():
    return 50

def test2(num):
    print(num)
#1、保存函数test1的返回值
result = test1()

#2、将函数返回值所在变量作为参数传递到test2函数
test2(result) # 50
```

16.4 函数的返回值

思考：如果一个函数如些两个 return（如下所示），程序如何执行?

```
def return_num():
    return 1
    return 2
result = return_num()
print(result)#1
```

答：只执行了第一个 return，原因是因为 return可以退出当前函数，导致 return下方的代码不执行。

思考：如果一个函数要有多个返回值，该如何书写代码?

```python
def return_num():
    return 1,2
    return  (10,20)
    return [100,200]
    return {'name':'python','age':30}

result = return_num()
print(result)#(1,2)是一个元组

```

注意：

1. `return a,b` 写法，返回多个数据的时候，默认是元组类型。
2. return后面可以连接列表、元组或字典，以返回多个值。

16.5 函数的参数

16.5.1 位置参数

位置参数：调用函数时根据函数定义的参数位置来传递参数。

```python
def users_info(name,age,gender):
    print(f'您的名字是{name},年龄是{age},性别是{gender}')
users_info('jjk',20,'男')
```

注意：传递和定义参数的顺序及个数必须一致

16.5.2 关键字参数

函数调用，通过“键=值”形式加以指定。可以让函数更加清晰、容易使用，同时也清除了参数的顺序需求。

```python
def users_info(name,age,gender):
    print(f'您的名字是{name},年龄是{age},性别是{gender}')
user_info('Rose',age=20,gender='女')
user_info('小敏',gender='男',age=20)
```

注意：函数调用时，如果有位置参数时，位置参数必须在关键字参数的前面但关键字参数之间不存在 先后顺序。

16.5.3 缺省参数

缺省参数也叫默认参数，用于定义函数，为参数提供默认值，调用函数时可不传该默认参数的值（注意：所有位置参数必须出现在默认参数前包括函数定义和调用）。

```python
def users_info(name,age,gender='男'):
    print(f'您的名字是{name},年龄是{age},性别是{gender}')

user_info('Rose',age=20)
user_info('小敏',age=20,gender='男')
```

注意：函数调用时，如果为缺省参数传值则修改默认参数值；否则使用这个默认值。

16.5.4 不定长参数

不定长参数也叫可变参数。用于不确定调用的时候会传递多少个参数（不传参也可以）的场景。此时， 用包裹（packing）位置参数，或者包裹关键字参数，来进行参数传递，会显得非常方便。

- **包裹位置传递**

```python
def user_info(*args):
    print(args)

#('tom',)
user_info('tom')
#('tom',18)
user_info('tom',18)
```

注意：传进的所有参数都会被args变量收集，它会根据传进参数的位置合并为一个元组（tuple） args是元组类型，这就是包裹位置传递。

- **包裹关键字传递**

```python
def user_info(**kwargs): # key word args
    print(kwargs)

# {'name': 'jjk', 'age': 18, 'id': 110}
user_info(name='jjk',age=18,id=110)
```

综上：无论是包裹位置传递还是包裹关键字传递，都是一个组包的过程。

16.6 拆包和交换变量值

16.6.1 拆包

- 拆包：元组

```python
def return_num():
    return 100,200 # 默认返回一个元组

num1,num2 = return_num()
print(num1) # 100
print(num2) # 200
```

- 拆包：字典

```python
dicts = {'name':'tom', 'age':18}
a, b =dict1
# 对字典进行拆包，取出来的是字典的key
print(a) # name
print(b) # age
print(dict1[a]) # tom
print(dict1[b]) # 18
```