day03

一、break和continue和range()

1. `break` 和 `continue` 是专门在循环中使用的关键字
2. `break` 某一条件满足，*不在执行循环中后续的代码*，并退出循环
3. `continue` 某一条件满足，*不在执行本次循环*，但会进入下次循环判断

```
"""---------------------- break ----------------------"""
box = ["酸奶", "积木", "可乐", "盒子", "矿泉水"]
for ele in box:
    print(ele)
    if ele == "可乐":
        break
# 结果:
酸奶
积木
可乐
"""--------------------- continue ---------------------"""
box = ["酸奶", "积木", "可乐", "盒子","可乐", "矿泉水"]
numb_f=0
for ele in box:
    if ele == "可乐":
        continue
    print(ele)
# 结果:
酸奶
积木
盒子
矿泉水
```

4. `range()`
   1. 可以创建一个整数列表
   2. 区间范围为 **左闭右开** `[4,5)`

```
"""---------------------- 区间的定义 ----------------------"""
```

```
# 从0开始可以省略,区间开始数据可以是负数
r = range(-1,10)
print(type(r))
for ele in r:
    print(ele,end=",")
# 结果:
<class 'range'>
-1,0,1,2,3,4,5,6,7,8,9,
"""---------------------- 打印偶数 ----------------------"""
for ele in range(0, 10, 2):
    print(ele, end=",")
# 结果:
0,2,4,6,8,
"""---------------------- 倒序区间 ----------------------"""
for ele in range(10,5,-1):
    print(ele, end=",")
# 结果:
10,9,8,7,6,
```

案例、机器人迎宾

```
"""---------------------- 机器人迎宾 ----------------------"""
from sdk.ur import UR

# 创建机器人
ur = UR()
# 连接机器人
ur.connect()
# 机器人姿态调整
ur.move_j((90, -90, 90, 180, -90, -90))

for ele in range(0,10):
    if ele %2 == 0:
        pass
        ur.move_l((0.10915, -0.36365, 0.83340, 180, -90, -90))
    else:
        ur.move_l((0.10915, -0.45830, 0.35885, 180, -90, -90))

ur.move_j((90, -90, 90, 180, -90, -90))
```

二、循环结合else

1. `for…else`格式

```
for 变量 in 集合:
    循环体代码
else:
    没有通过 break 退出循环，循环结束后，会执行的代码
```

2. `else` 中的语句会在**循环正常执行完**（即 for 不是通过 break 跳出而中断的）的情况下**执行**,while循环也一样

```
"""-------------------- for和else --------------------"""
box = ["酸奶", "积木", "可乐", "盒子", "可乐", "矿泉水"]
numb_f = 0
for ele in box:
    if ele == "可乐":
        continue
    print(ele)
else:
    print("执行了！！！")
# 结果：
酸奶
积木
盒子
矿泉水
执行了！！！
"""-------------------- for和else --------------------"""
box = ["酸奶", "积木", "可乐", "盒子", "可乐", "矿泉水"]
numb_f = 0
for ele in box:
    if ele == "可乐":
        break
    print(ele)
else:
    print("执行了！！！")
# 结果：
酸奶
积木
```

三、元组tuple

1. 元组的定义用 `()`，元素之间用 `,` 隔开，元素索引从 `0` 开始，元组类型 `tuple`
2. **注意：当元组中只有一个元素的时候，必须在后面加 `,`**
3. 元组特点：*元素不能修改*

```
"""---------------------- 元组 ----------------------"""
t = ("积木", "酸奶", "可乐", "盒子")
print(type(t))
print(t)
# 结果：
<class 'tuple'>
('积木', '酸奶', '可乐', '盒子')
```

4. 元组常见操作：

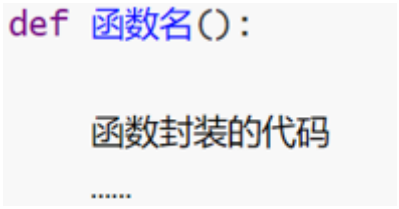
分类	方法	说明
查询	`元组[索引]`	根据索引值，索引不存在会报错
	`index(数据)`	根据查询索引，返回首次出现的所有，没有查到会报错
	`count(数据)`	数据在元素中出现的次数
	`len(元组)`	元组长度
	`if 数据 in 元组:`	检查元组中是否包含某元素
遍历	`for 元素 in 元组:`	取出元组中的每一个元素

5. 使用场景
    - 自动组包
    - 自动解包
    - 交换数据
    - 格式化输出
    - 让列表布局修改，保护数据安全

```python
"""---------------------- 元组使用场景 ----------------------"""
# 自动组包
t = 1, 2, 3
print(t)
# 自动解包 需要数量对应，才能解包
a, b, c = t
print(a, b, c)
# 交换数据
a, b = b, a
print(a, b, c)
# 格式化输出
name = "刘小刚"
age = 10
print("名字是%s,年纪是%d" % (name, age))
# 保证数据安全
list1 = [10,20]
tuple1 = tuple(list1)
print(tuple1)
# 结果：
(1, 2, 3)
1 2 3
2 1 3
名字是刘小刚,年纪是10
(10, 20)
```

四、函数

1. 函数是计算机执行命令的单元
2. 函数的定义格式



3. 函数调用格式：`函数名()`

```
"""---------------------- 函数 ----------------------"""
# 函数定义
def crawl():
    print("3D相机扫描确定物品位置")
    print("移动物品")
# 函数调用
crawl()
# 结果:
3D相机扫描确定物品位置
移动物品
```

4. 函数的文档注释

```
"""---------------------- 函数 ----------------------"""
def crawl():
    """
    这是一个抓取的功能
    """
    print("3D相机扫描确定物品位置")
    print("移动物品")

crawl()
# Ctrl+Q 查看文档注释
```

5. 函数参数的定义格式


```
"""---------------------- 函数返回值 ----------------------"""
def Sum(a,b):
    return a+b

print(Sum(10, 10))
# 结果:
20
```

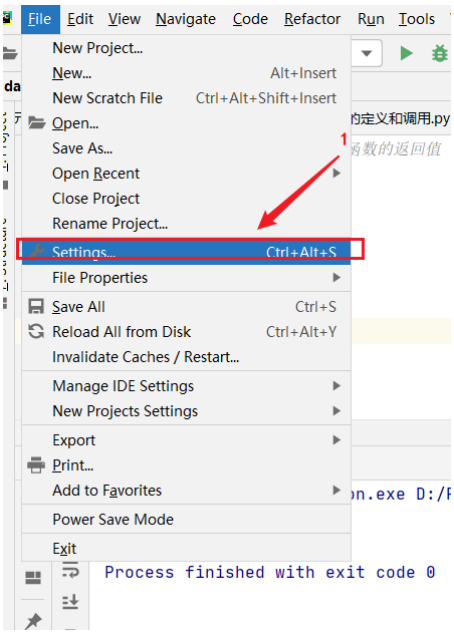
6. 返回值：是函数给调用方提供的结果
7. 返回值可以是多个

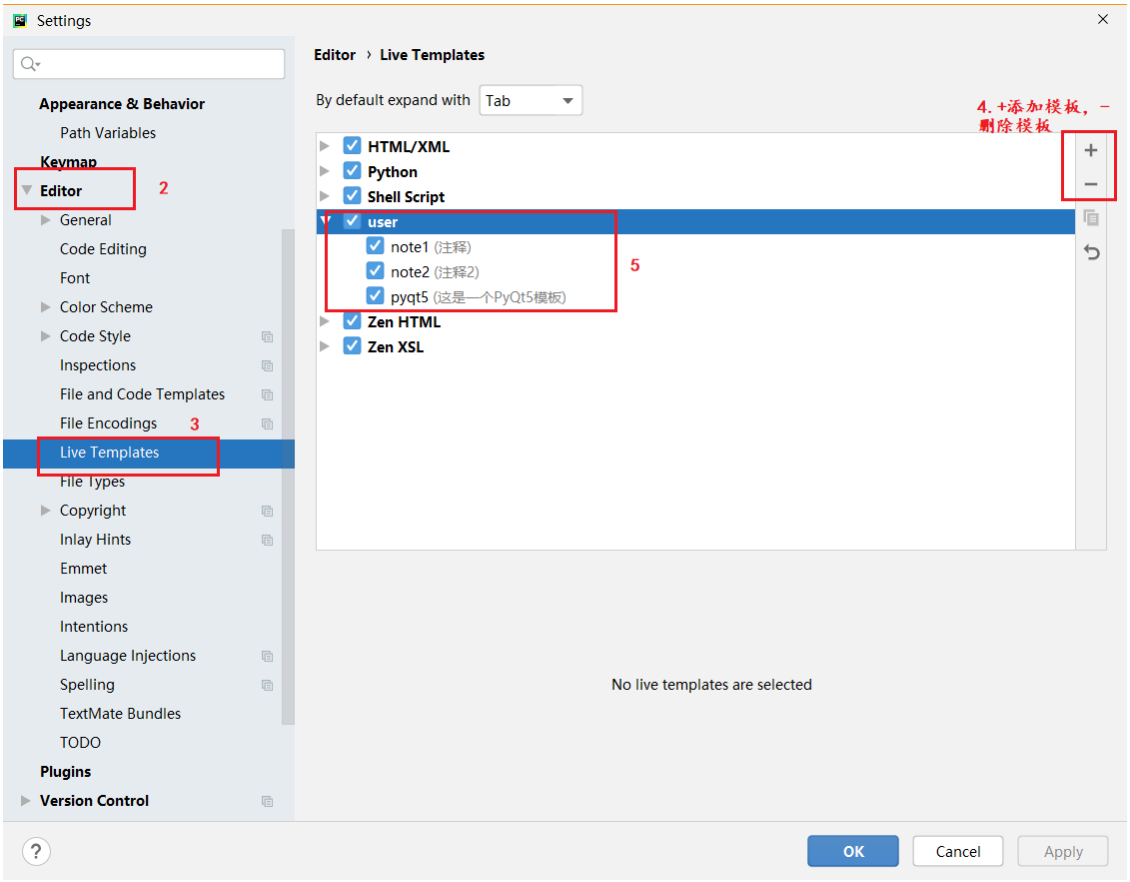
```python
"""---------------------- 函数的返回值 ----------------------"""
def myMax(a, b):
    return a + b, a - b
num = myMax(20, 10)
print(num)
```

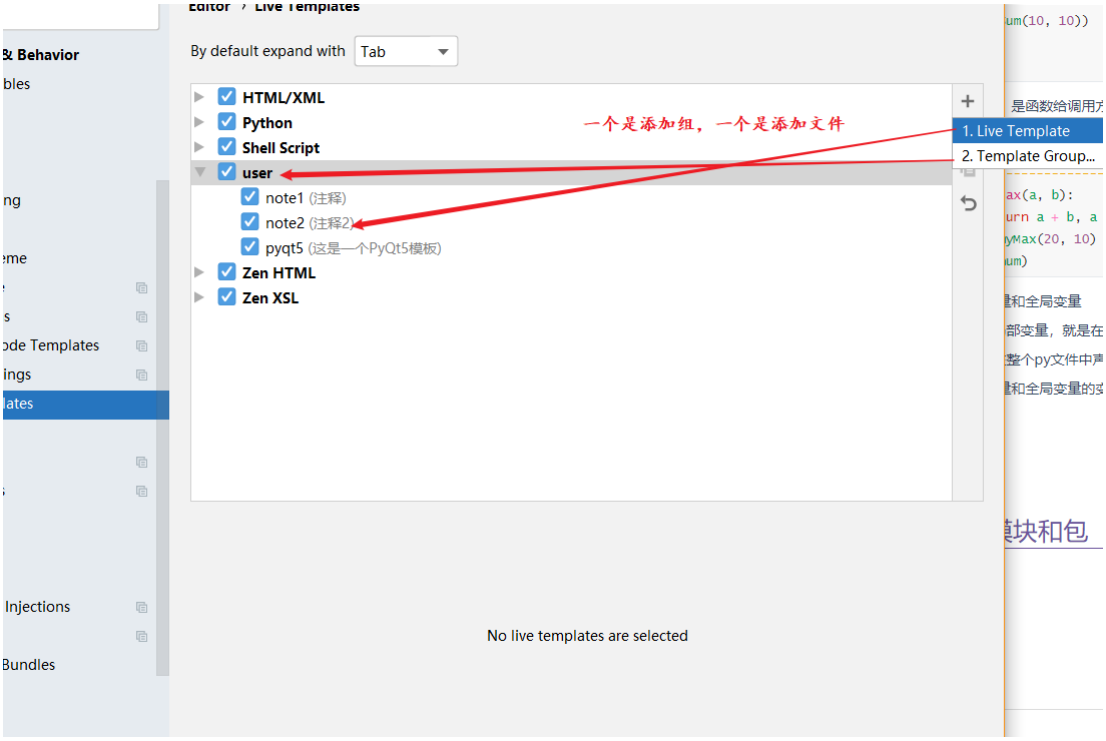
8. **局部变量和全局变量**
    - 局部变量，就是在函数内部定义的变量、只能在函数内部调用
    - 在整个py文件中声明，全局范围内都可以访问
9. 局部变量和全局变量的变量名相同时，修改的就是局部变量值，想修改全局变量值就得用 `global`

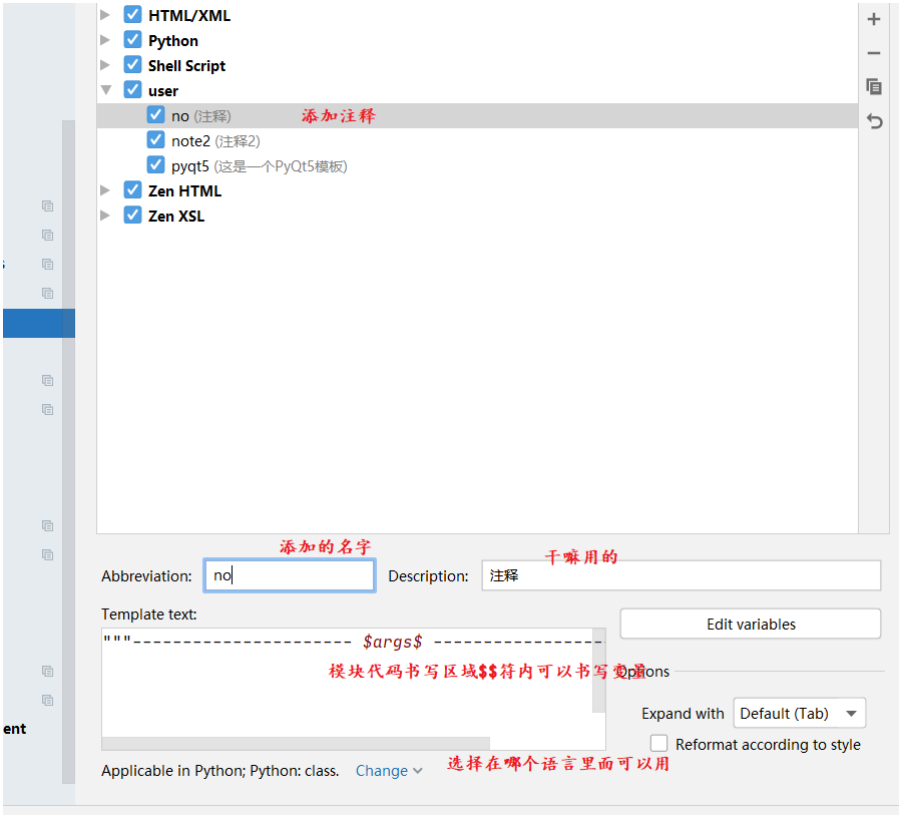
```python
"""---------------------- 局部变量和全局变量 ----------------------"""
num = 10
print(num)
def add():
    a = 10
    global num
    print(a,num)
    num = 30
add()
print(num)
# 结果：
10
10 10
30
```

模板配置




Settings
Editor › Live Templates
By default expand with Tab
4. +添加模板，-删除模板
Appearance & Behavior
Path Variables
Keymap
Editor
2
General
Code Editing
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
3
Live Templates
File Types
Copyright
Inlay Hints
Emmet
Images
Intentions
Language Injections
Spelling
TextMate Bundles
TODO
Plugins
Version Control
HTML/XML
Python
Shell Script
user
note1 (注释)
note2 (注释2)
pyqt5 (这是一个PyQt5模板)
5
Zen HTML
Zen XSL
No live templates are selected
OK
Cancel
Apply


By default expand with Tab
HTML/XML
Python
Shell Script
user
note1 (注释)
note2 (注释2)
pyqt5 (这是一个PyQt5模板)
Zen HTML
Zen XSL
一个是添加组，一个是添加文件
1. Live Template
2. Template Group...
No live templates are selected



五、模块和包

1. 模块就像工具包，每一个py文件就是一个模块
2. 用 `import` 和 `from...import...` 导入，可以用 `as` 对导入的模块起别名