1. 今日内容大纲

1. 如何在工作中不让别人看出你是培训出来的?
    - 第一天环境安装等等，小白各种问。
    - 项目需求不清晰，也不敢问。
    - 我们6个月一定要学会自主学习，自己解决问题的能力。
2. 形参角度：
    - 万能参数。
    - *的魔性用法。
    - 仅限关键字参数（了解）。
    - 形参的最终顺序。
3. 名称空间。
    1. 全局名称空间，局部.....
    2. 加载顺序，取值顺序。
    3. 作用域。
4. 函数的嵌套（高阶函数）。
5. 内置函数 globals locals
6. 关键字：nonlocal global。

2. 昨日内容回顾作业讲解

- 函数是以功能为导向，减少重复代码，提高代码的可读性。

```
def func():
    函数体
```

- 函数的调用：func（）

```
func()
func()
func()
```

- 函数的返回值 return
    - 终止函数。
    - return 单个值：
    - return 多个值：(1,2,3,'alex')
- 函数的参数：
    - 实参角度：位置参数，关键字参数，混合参数。
    - 形参角度：位置参数，默认参数。

3. 今日内容

1. 如何在工作中不让别人看出你是培训出来的?
    - 第一天环境安装等等，小白各种问。
    - 项目需求不清晰，也不敢问。
    - 我们6个月一定要学会自主学习，自己解决问题的能力。
2. 形参角度:
    - 万能参数。
    - *的魔性用法。

```
# 万能参数。
# def eat(a,b,c,d):
#     print('我请你吃: %s,%s,%s,%s' %(a,b,c,d))
#
# eat('蒸羊羔', '蒸熊掌', '蒸鹿邑','烧花鸭')

# def eat(a,b,c,d,e,f):
#     print('我请你吃: %s,%s,%s,%s,%s,%s' %(a,b,c,d,e,f))
#
# eat('蒸羊羔', '蒸熊掌', '蒸鹿邑','烧花鸭','烧雏鸡','烧子鹅')

# 急需要一种形参，可以接受所有的实参。#万能参数。
# 万能参数:  *args, 约定俗称: args,
# 函数定义时，*代表聚合。 他将所有的位置参数聚合成一个元组，赋值给了 args。

# def eat(*args):
#     print(args)
#     print('我请你吃: %s,%s,%s,%s,%s,%s' % args)
#
# eat('蒸羊羔', '蒸熊掌', '蒸鹿邑','烧花鸭','烧雏鸡','烧子鹅')

# 写一个函数: 计算你传入函数的所有的数字的和。
# def func(*args):
#     count = 0
#     for i in args:
#         count += i
#     return count
# print(func(1,2,3,4,5,6,7))

# tu1 = (1, 2, 3, 4, 5, 6, 7)
# count = 0
# for i in tu1:
#     count += i
# print(count)

# **kwargs
# 函数的定义时:  ** 将所有的关键字参数聚合到一个字典中，将这个字典赋值给了kwargs.
# def func(**kwargs):
#     print(kwargs)
# func(name='alex',age=73,sex='laddyboy')
```

```
# 万能参数: *args, **kwargs,
# def func(*args,**kwargs):
#     print(args)
#     print(kwargs)
# func()
# print()

# * **在函数的调用时，*代表打散。
def func(*args,**kwargs):
    print(args) # (1,2,3,22,33)
    print(kwargs)

# func(*[1,2,3],*[22,33])  # func(1,2,3,22,33)
# func(*'fjskdfsa',*'fkjdsal')  # func(1,2,3,22,33)
func(**{'name': '太白'},**{'age': 18})  #func(name='太白',age='18')
```

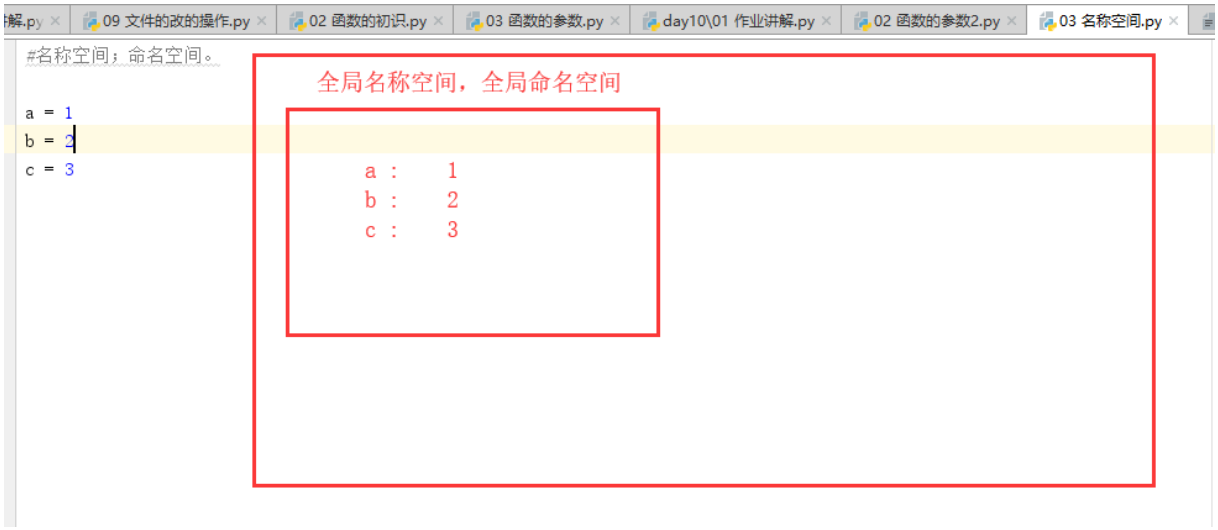
- 仅限关键字参数（了解）。
- 形参的最终顺序。

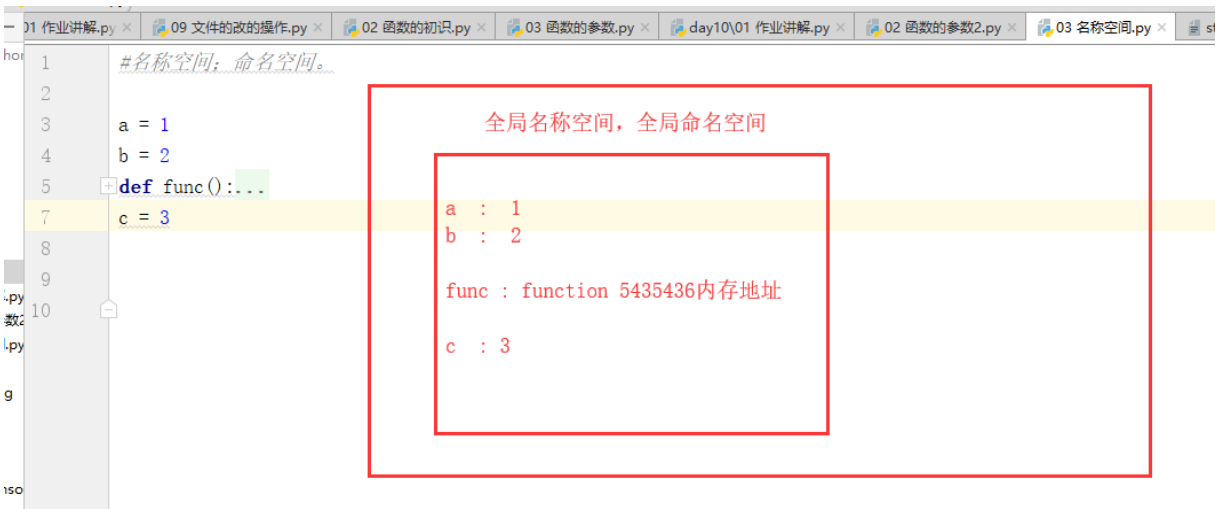
```
# 形参角度的参数的顺序
# *args 的位置?
# def func(*args,a,b,sex= '男'):
#     print(a,b)
# func(1,2,3,4)
# args得到实参的前提，sex必须被覆盖了。
# def func(a,b,sex= '男',*args,):
#     print(a,b)
#     print(sex)
#     print(args)
# func(1,2,3,4,5,6,7,)

# def func(a,b,*args,sex= '男'):
#     print(a,b)
#     print(sex)
#     print(args)
# func(1,2,3,4,5,6,7,sex='女')

# **kwargs 位置?
def func(a,b,*args,sex= '男',**kwargs,):
    print(a,b)
    print(sex)
    print(args)
    print(kwargs)
# func(1,2,3,4,5,6,7,sex='女',name='Alex',age=80)

# 形参角度第四个参数: 仅限关键字参数 (了解)

def func(a,b,*args,sex= '男',c,**kwargs,):
    print(a,b)
    print(sex)
```

```
    print(args)
    print(c)
    print(kwargs)
# func(1,2,3,4,5,6,7,sex='女',name='Alex',age=80,c='666')

# 形参角度最终的顺序：位置参数,*args,默认参数，仅限关键字参数，**kwargs
```

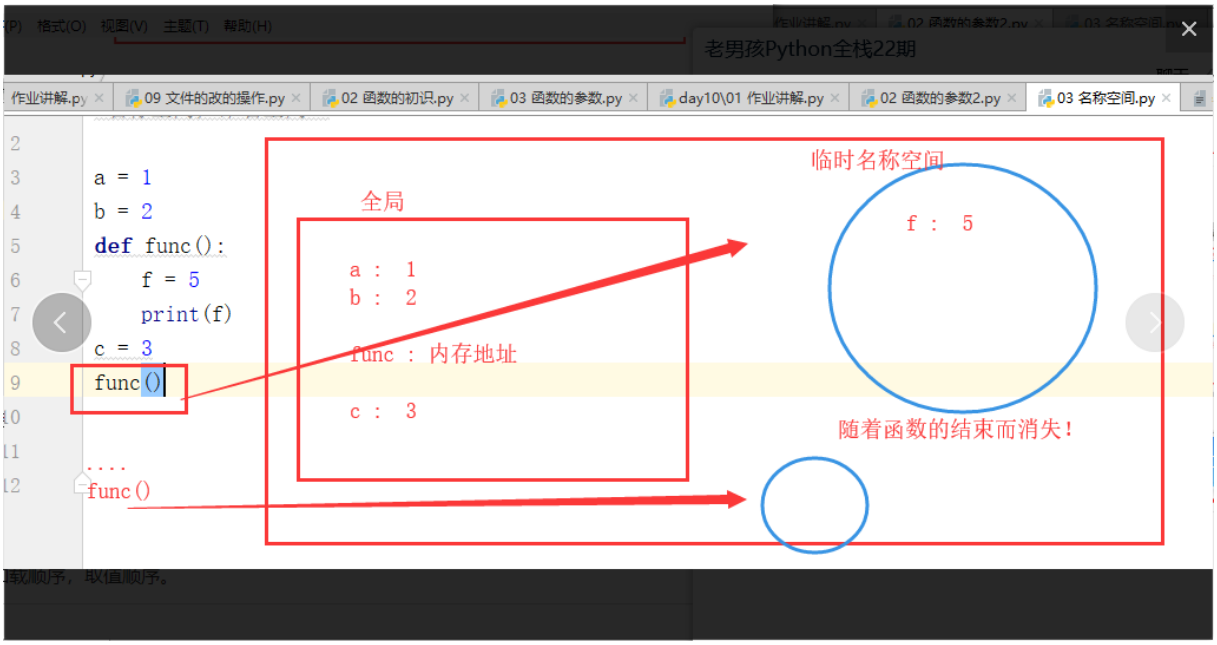
3. 名称空间。

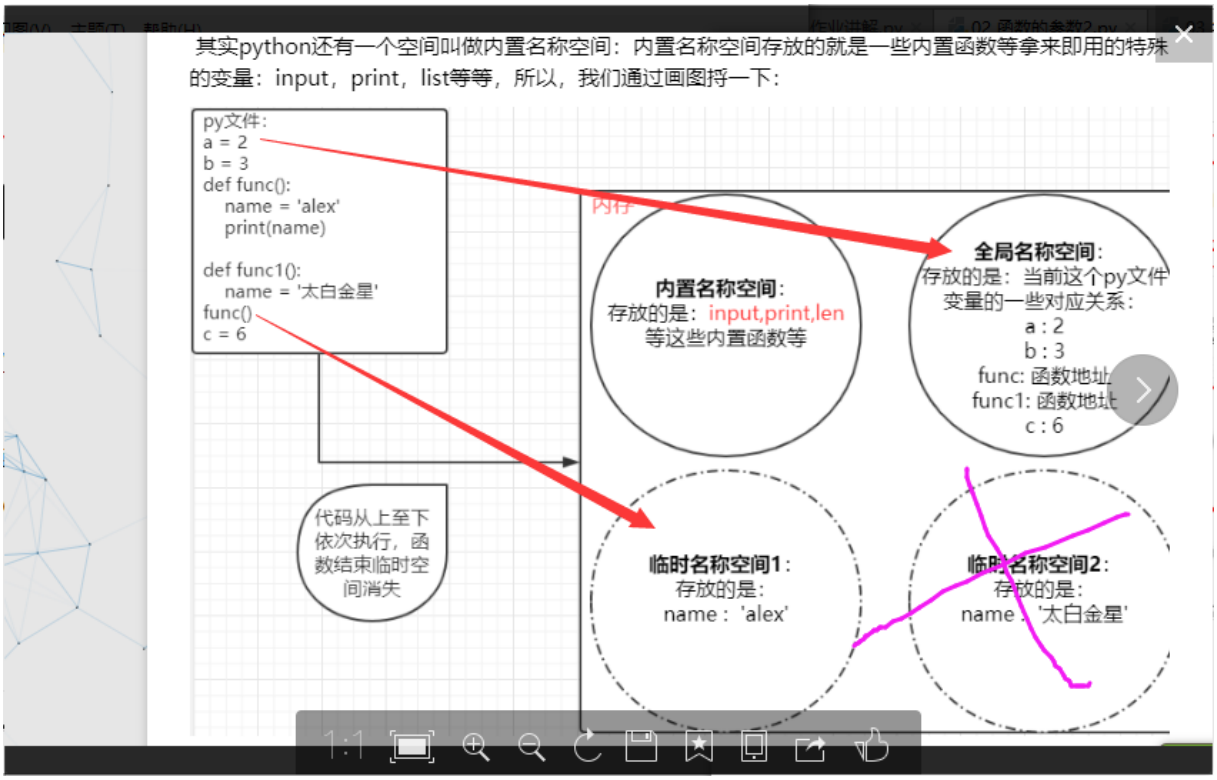
    1. 全局名称空间，局部.....



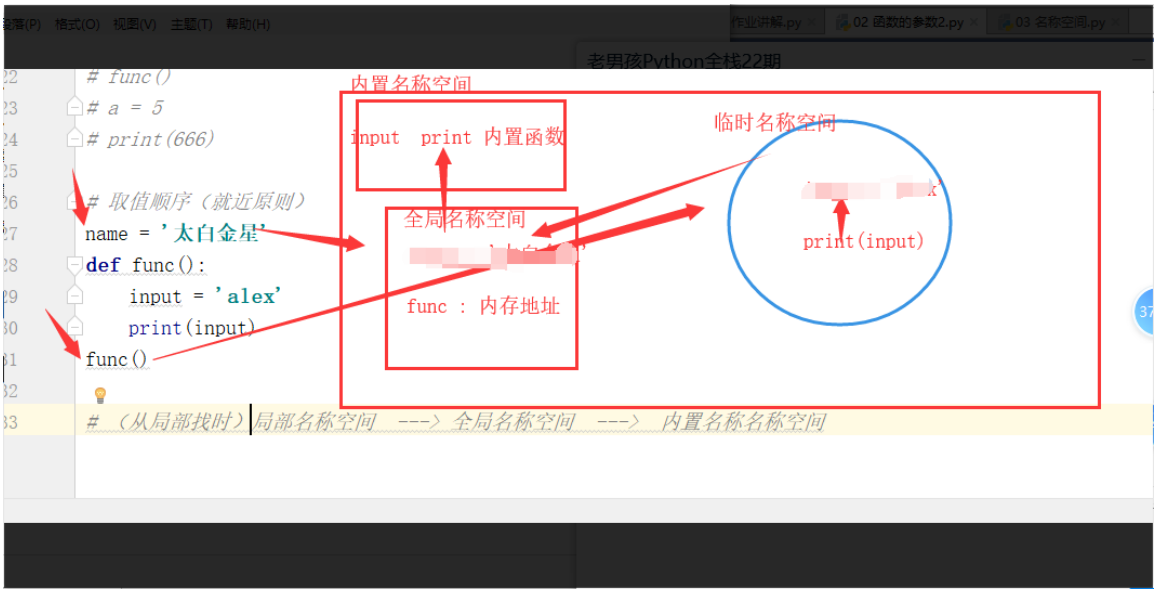


局部名称空间（临时名称空间）：


老男孩Python全栈22期
作业讲解.py
09 文件的改的操作.py
02 函数的初识.py
03 函数的参数.py
day10\01 作业讲解.py
02 函数的参数2.py
03 名称空间.py
a = 1
b = 2
def func():
f = 5
print(f)
c = 3
func()
....
func()
全局
a : 1
b : 2
func : 内存地址
c : 3
临时名称空间
f : 5
随着函数的结束而消失!


其实python还有一个空间叫做内置名称空间：内置名称空间存放的就是一些内置函数等拿来即用的特殊的变量：input，print，list等等，所以，我们通过画图捋一下：
py文件:
a = 2
b = 3
def func():
name = 'alex'
print(name)
def func1():
name = '太白金星'
func()
c = 6
内存
内置名称空间:
存放的是：input,print,len等这些内置函数等
全局名称空间:
存放的是：当前这个py文件
变量的一些对应关系：
a : 2
b : 3
func: 函数地址
func1: 函数地址
c : 6
代码从上至下依次执行，函数结束临时空间消失
临时名称空间1:
存放的是:
name : 'alex'
临时名称空间2:
存放的是:
name : '太白金星'

1. 加载顺序，取值顺序。




```
#名称空间；命名空间。

# a = 1
# b = 2
# def func():
#     f = 5
#     print(f)
# c = 3
# func()

# 内置名称空间：python源码给你提供的一些内置的函数，print input
# print(666)
# python分为三个空间：
    # 内置名称空间（builtins.py）
    # 全局名称空间（当前py文件）
    # 局部名称空间（函数，函数执行时才开辟）

# 加载顺序：
# 内置名称空间 ---> 全局名称空间  ----> 局部名称空间（函数执行时）
# def func():
#     pass
# func()
# a = 5
# print(666)
python
```

```
# 加载顺序：
# 内置名称空间 ---> 全局名称空间  ----> 局部名称空间（函数执行时）
# def func():
#     pass
# func()
# a = 5
# print(666)

# 取值顺序（就近原则）  单向不可逆
```

```
# LEGB原则
# input = '太白金星'
def func():
    # input = 'alex'
    print(input)
# func()

# （从局部找时）局部名称空间  ---> 全局名称空间  --->  内置名称名称空间


# input = '太白金星'
# def func():
#     input = 'alex'
#
# func()
# print(input)
```

2. 作用域。

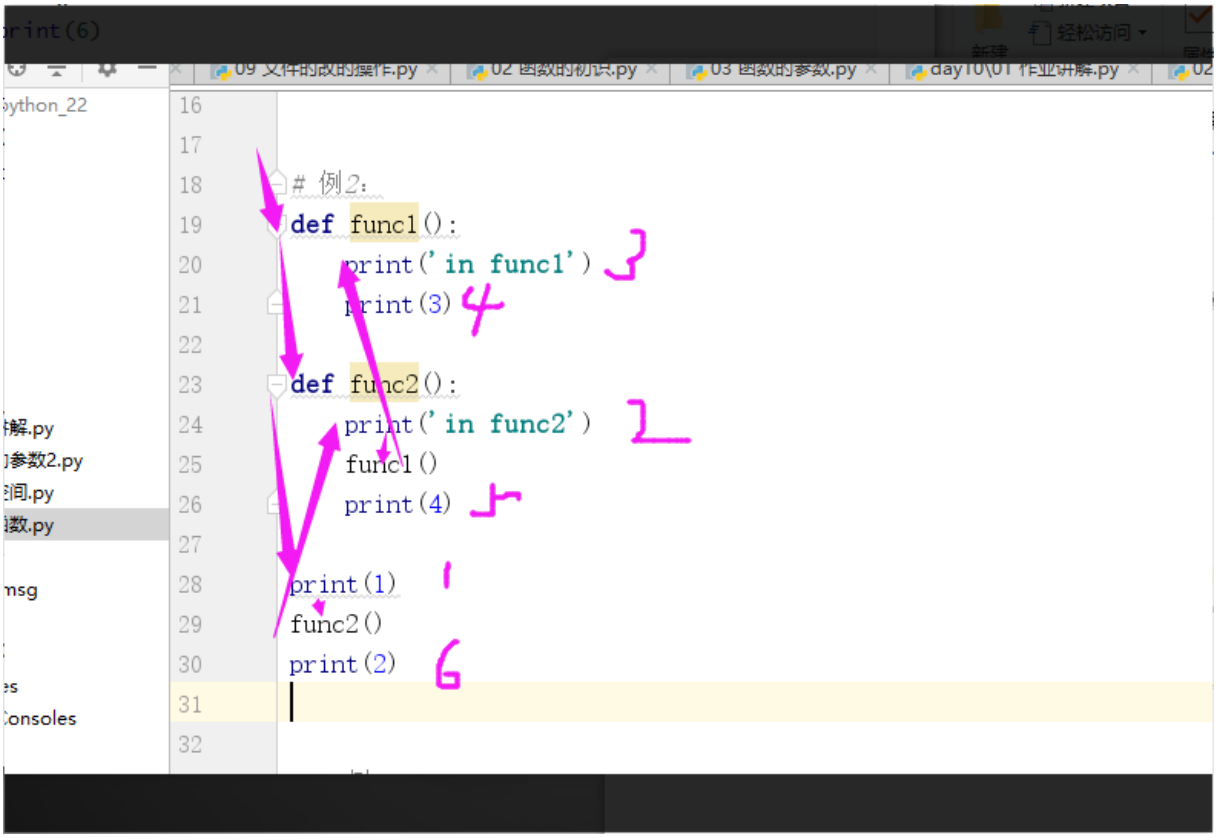
```
# 作用域:
# 两个作用域:
    # 全局作用域 ： 内置名称空间 全局名称空间
    # 局部作用域: 局部名称空间

#  局部作用域可以引用全局作用域的变量
# date = '周五'
# def func():
#     a = 666
#     print(date)
# print(a)
# func()
# print(a)

# 局部作用域不能改变全局变量。
# count = 1
# # def func():
# #     count += 2
# #     print(count)
# # func()  # local variable 'count' referenced before assignment
# 局部作用域不能改变全局作用域的变量，当python解释器读取到局部作用域时，发现了你对一个变量进行修改的操作，
# 解释器会认为你在局部已经定义过这个局部变量了，他就从局部找这个局部变量，报错了。

# 使用可以，不能改变
# def func():
#     count = 1
#     def inner():
#         print(count)
#     inner()
# func()
```

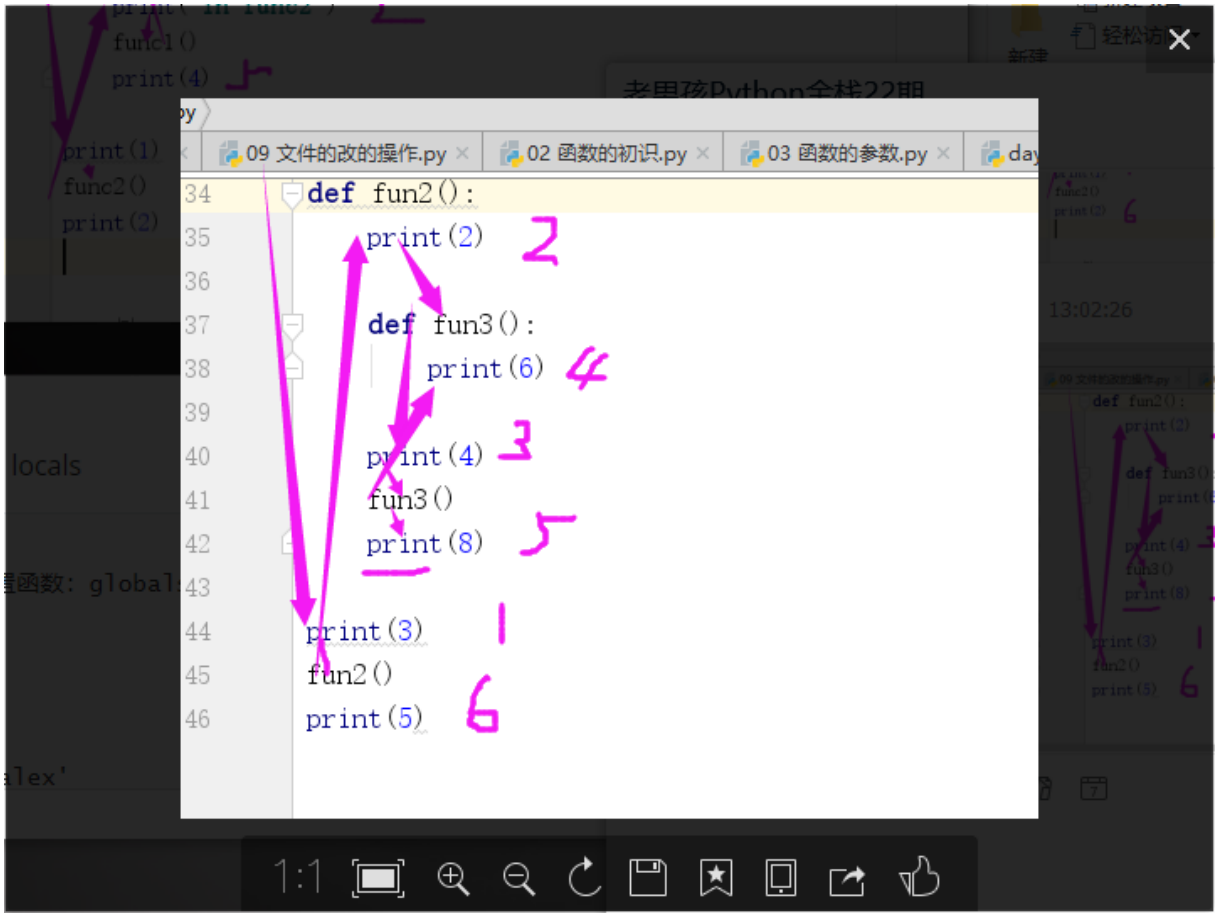
```
def func():
    count = 1
    def inner():
        count += 1
        print(count)
    inner()
func()
```

4. 函数的嵌套（高阶函数）。

```
# 例1:
def func1():
    print('in func1')
    print(3)

def func2():
    print('in func2')
    print(4)

func1()
print(1)
func2()
print(2)
# in func1 3 1 in func2 4 2



# 例2:
def func1():
    print('in func1')
    print(3)

def func2():
    print('in func2')
    func1()
    print(4)

print(1)
func2()
print(2)


# 例3:
def fun2():
    print(2)

    def fun3():
        print(6)

    print(4)
```

```
    fun3()
    print(8)

print(3)
fun2()
print(5)


# glbals() locals()
```





5. 内置函数 globals locals

```
"""
本文件：研究内置函数：globals locals
"""
a = 1
b = 2
def func():
    name = 'alex'
    age = 73
    print(globals())  # 返回的是字典：字典里面的键值对：全局作用域的所有内容。
    print(locals())  # 返回的是字典：字典里面的键值对：当前作用域的所有的内容。
# print(globals())  # 返回的是字典：字典里面的键值对：全局作用域的所有内容。
# print(locals())  # # 返回的是字典：字典里面的键值对：当前作用域的所有的内容。
func()
```

6. 关键字：nonlocal global。

4. 今日总结

1. 参数：万能参数，仅限关键字参数（了解）,参数的顺序，*的魔性用法：聚合，打散。
2. 名称空间，作用域，取值顺序，加载顺序。
3. globals locals
4. 高阶函数：执行顺序。

5. 以上全部都是重点。

5. 下周预习内容

1. 下周一 函数名的运用，迭代器。