

## 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: ** 将所有关键字参数聚合成一个字典中，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','fkjdsa1') # 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.....

The top screenshot shows a Python IDE with a file named '03 名称空间.py'. The code in the editor is:

```

#名称空间; 命名空间。
a = 1
b = 2
c = 3

```

The output window shows the following content, indicating the global namespace:

```

全局名称空间, 全局命名空间

a : 1
b : 2
c : 3

```

The bottom screenshot shows the same IDE with a function definition added to the code:

```

1 #名称空间; 命名空间。
2
3 a = 1
4 b = 2
5 def func():...
6
7 c = 3
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150

```

The output window shows the following content, indicating the global namespace and the function object:

```

全局名称空间, 全局命名空间

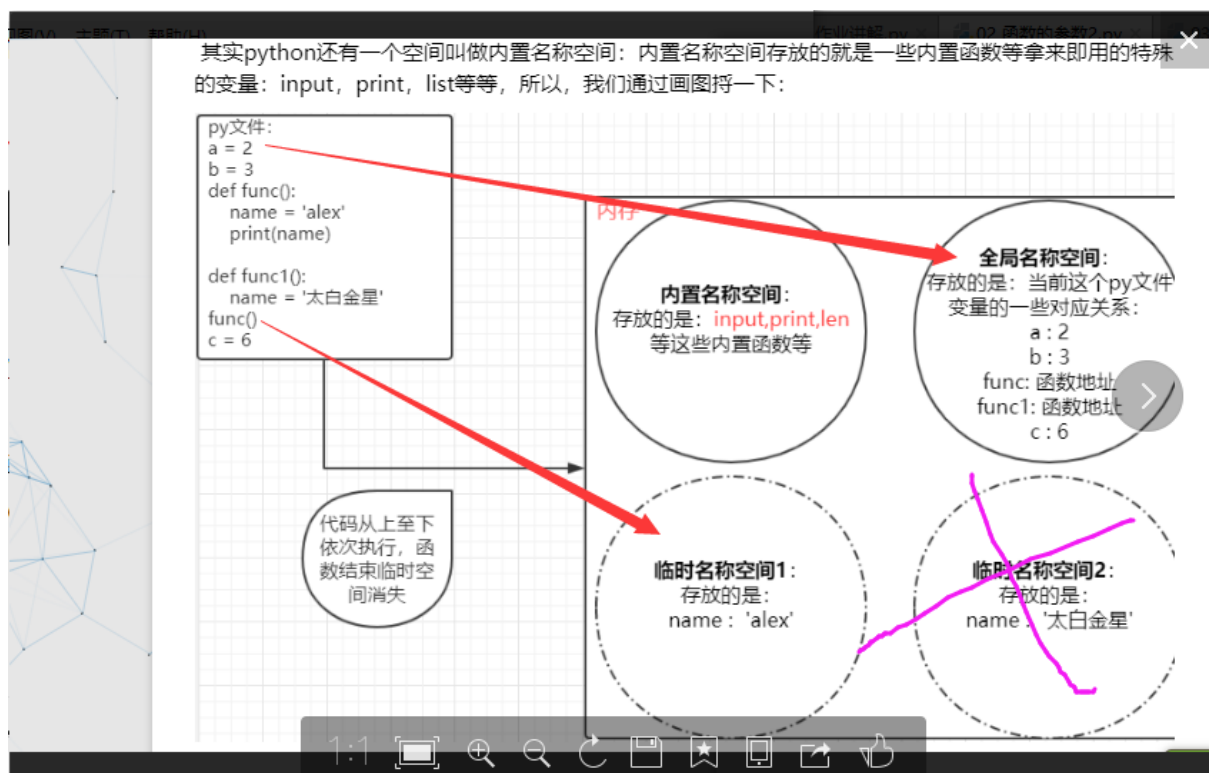
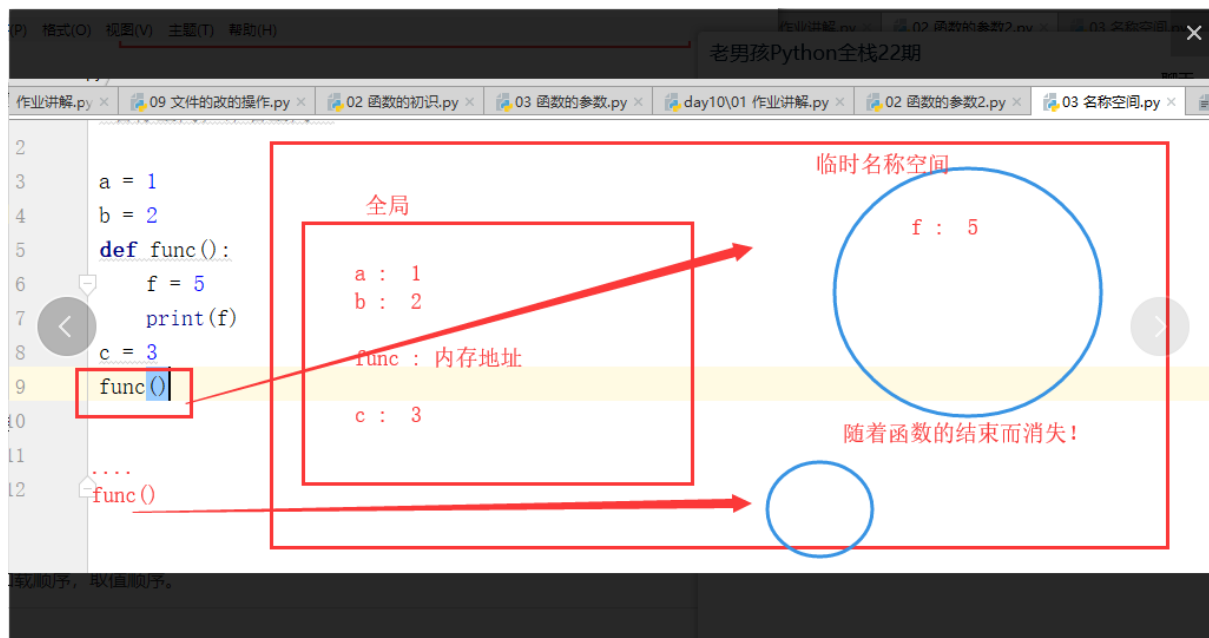
a : 1
b : 2

func : function 5435436内存地址

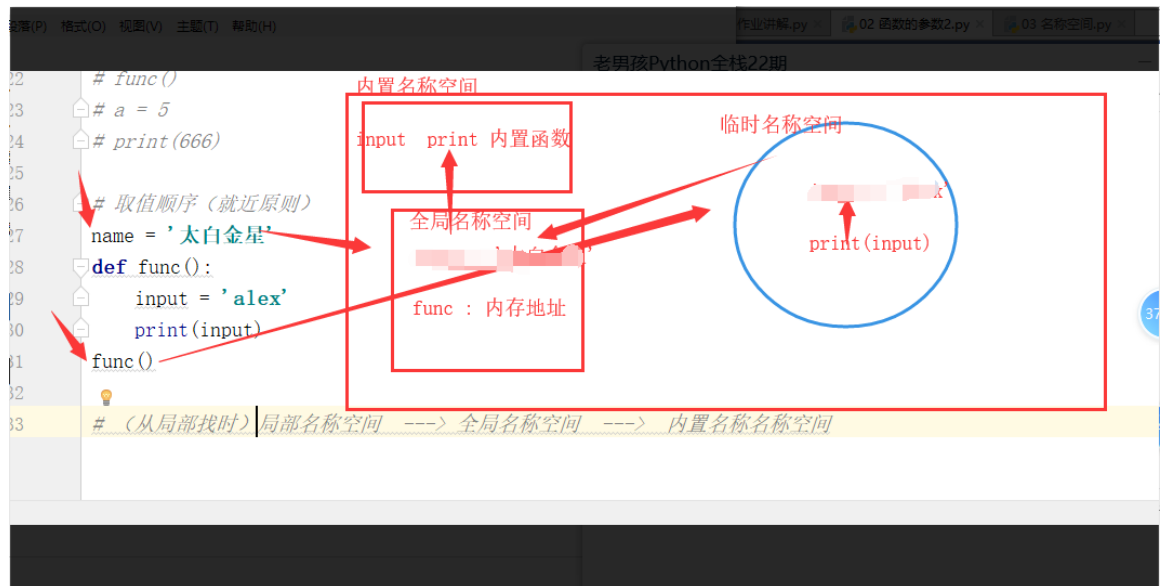
c : 3

```

局部名称空间 (临时名称空间) :



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()
```

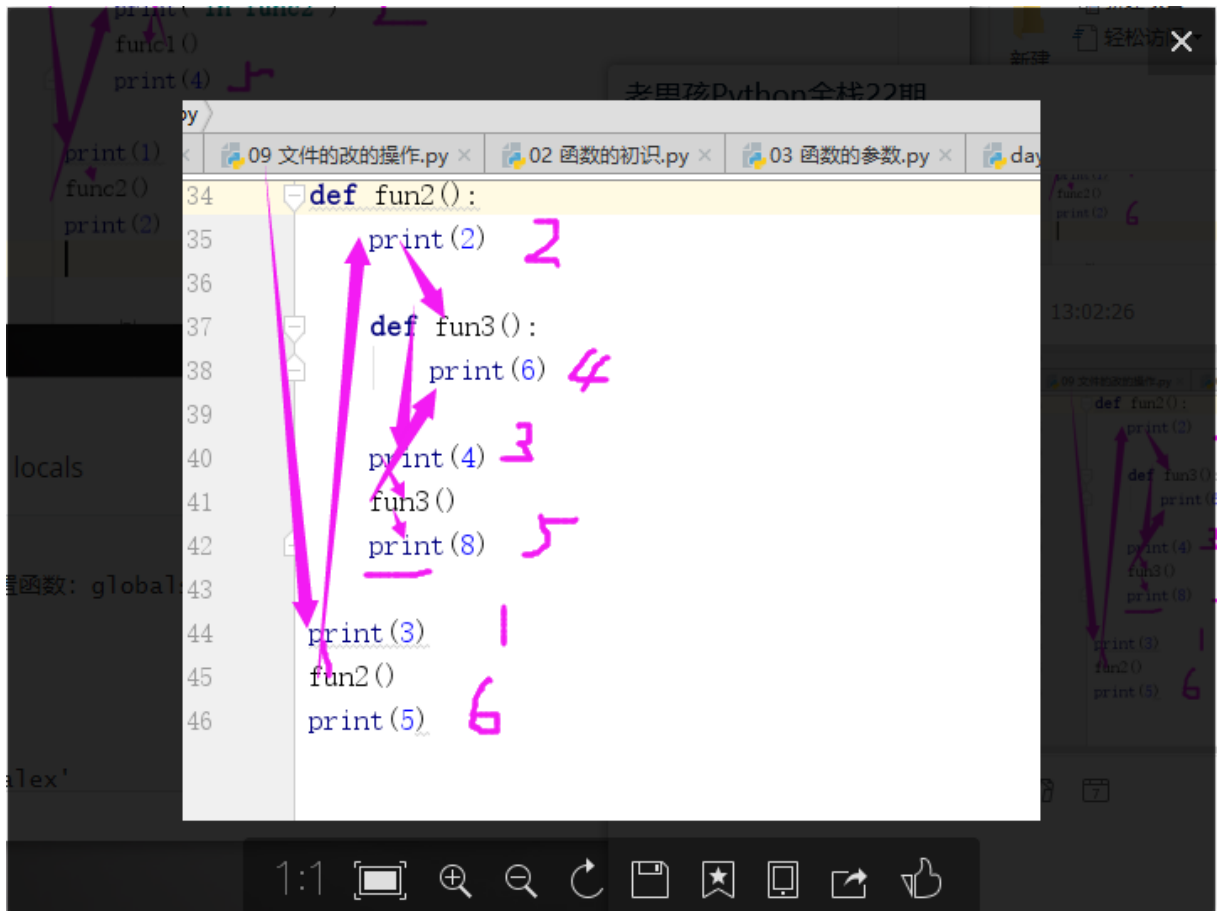
The screenshot shows a Python IDE with a file explorer on the left and a code editor. The code editor contains the following Python code:

```
16
17
18 # 例2:
19 def func1():
20     print('in func1')
21     print(3)
22
23 def func2():
24     print('in func2')
25     func1()
26     print(4)
27
28 print(1)
29 func2()
30 print(2)
31
32
```

Handwritten annotations in pink include:

- Arrows pointing from the function definitions to their calls: from line 19 to line 25, from line 23 to line 25, and from line 29 to line 25.
- Curly braces on the right side of the code, grouping lines: a brace for lines 20-21, a brace for lines 24-26, and a brace for lines 28-30.
- Numbers written next to specific lines: '4' next to line 21, '2' next to line 24, '1' next to line 28, and '6' next to line 30.

The file explorer on the left shows a project named 'python\_22' with files: '解.py', '参数2.py', '同.py', and '数.py'. The '数.py' file is selected.



#### 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。