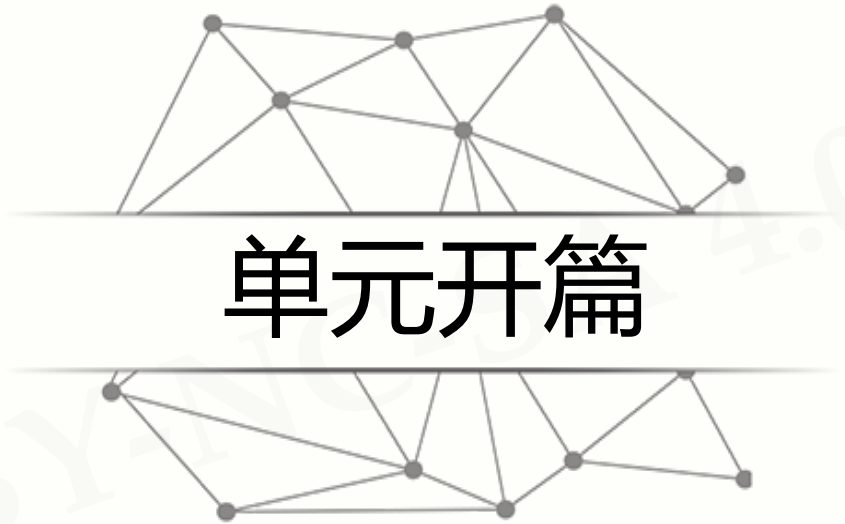

python
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
turtle.seth(-40)
for 1 in range(4):
turtle.circle(40, 80)
turtle.circle(-40, 80)
turtle.circle(40, 80/2)
turtle.fd(40)
turtle.circle(16, 180)
turtle.fd(40 * 2/3)

Python语言程序设计

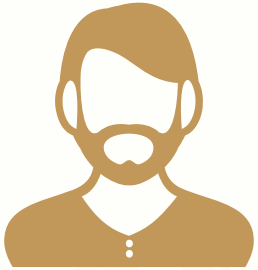
函数的定义与使用

嵩 天
北京理工大学

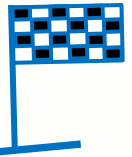


单元开篇

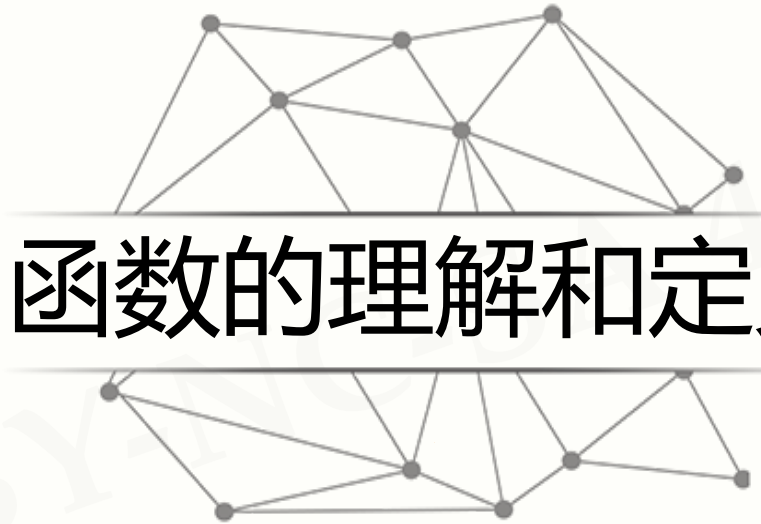
函数的定义与使用



- 函数的理解与定义
- 函数的使用及调用过程
- 函数的参数传递
- 函数的返回值
- 局部变量和全局变量
- lambda函数



python
语言程序设计



函数的理解和定义

#DayDayUpQ4.py

```
def dayUP(df):
    dayup = 1
    for i in range(365):
        if i % 7 in [6,0]:
            dayup = dayup*(1 - 0.01)
        else:
            dayup = dayup*(1 + df)
    return dayup
dayfactor = 0.01
while dayUP(dayfactor) < 37.78:
    dayfactor += 0.001
print("工作日的努力参数是：{:.3f}".format(dayfactor))
```

def..while..

("笨办法"试错)

python

函数的定义

函数是一段代码的表示

- 函数是一段具有特定功能的、可重用的语句组
- 函数是一种功能的抽象，一般函数表达特定功能
- 两个作用：降低编程难度 和 代码复用

函数的定义

函数是一段代码的表示

```
def   <函数名>(<参数(0个或多个)>) :
      <函数体>
      return   <返回值>
```

函数的定义

计算 n!

函数名

参数

```
def fact(n) :
    s = 1
    for i in range(1, n+1):
        s *= i
    return s
```

返回值

函数的定义

$$y = f(x)$$

- 函数定义时，所指定的参数是一种**占位符**
- 函数定义后，如果不经过**调用**，不会被执行
- 函数定义时，参数是输入、函数体是处理、结果是输出 (IPO)

函数的使用及调用过程

函数的调用

调用是运行函数代码的方式

```
def fact(n) :
    s = 1
    for i in range(1, n+1):
        s *= i
    return s
```

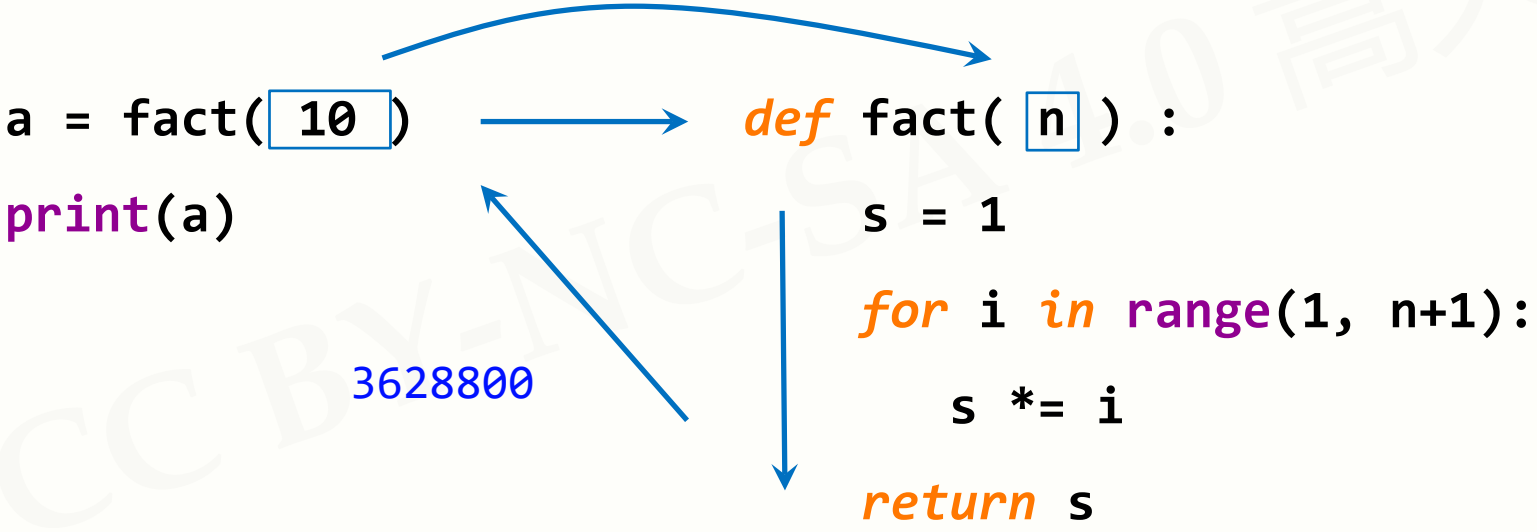
函数的定义

```
fact(10)
```

函数的调用

- 调用时要给出实际参数
- 实际参数替换定义中的参数
- 函数调用后得到返回值

函数的调用过程



函数的参数传递

参数个数

函数可以有参数，也可以没有，但必须保留括号

```
def   <函数名>() :
          <函数体>
          return   <返回值>
```

```
def fact() :
    print("我也是函数")
```

可选参数传递

函数定义时可以为某些参数指定默认值，构成可选参数

```
def   <函数名>(<非可选参数>，<可选参数>) :
      <函数体>
      return   <返回值>
```

可选参数传递

计算 n!//m

可选参数

```
def fact(n, m=1) :
    s = 1
    for i in range(1, n+1):
        s *= i
    return s//m
```

```
>>> fact(10)
3628800
>>> fact(10,5)
725760
```

可变参数传递

函数定义时可以设计可变数量参数，既不确定参数总数量

```
def   <函数名>(<参数>, *b ) :
        <函数体>
        return   <返回值>
```

可变参数传递

计算 n!乘数

可变参数

```
def fact(n, *b) :
    s = 1
    for i in range(1, n+1):
        s *= i
    for item in b:
        s *= item
    return s
```

```
>>> fact(10,3)
10886400
>>> fact(10,3,5,8)
435456000
```

参数传递的两种方式

函数调用时，参数可以按照位置或名称方式传递

```
def fact(n, m=1) :
    s = 1
    for i in range(1, n+1):
        s *= i
    return s//m
```

位置传递

```
>>> fact( 10,5 )
725760
```

名称传递

```
>>> fact( m=5,n=10 )
725760
```

函数的返回值

函数的返回值

函数可以返回0个或多个结果

- `return`保留字用来传递返回值
- 函数可以有返回值，也可以没有，可以有`return`，也可以没有
- `return`可以传递0个返回值，也可以传递任意多个返回值

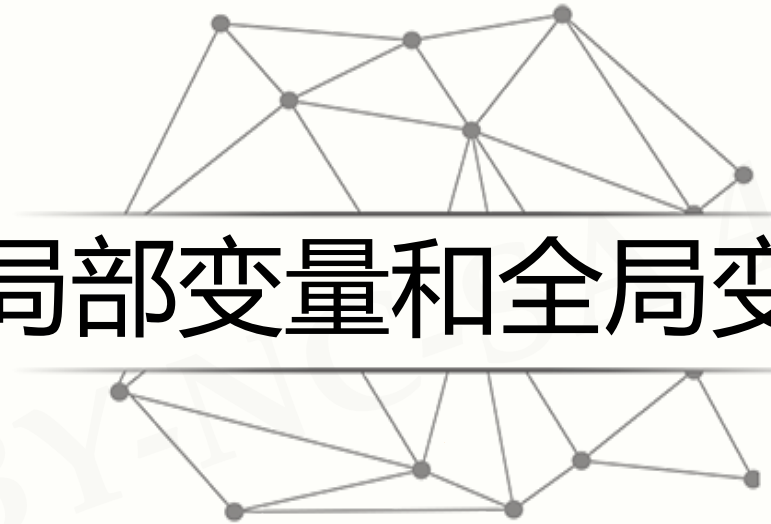
函数的返回值

函数调用时，参数可以按照位置或名称方式传递

```
def fact(n, m=1) :
    s = 1
    for i in range(1, n+1):
        s *= i
    return s//m, n, m
```

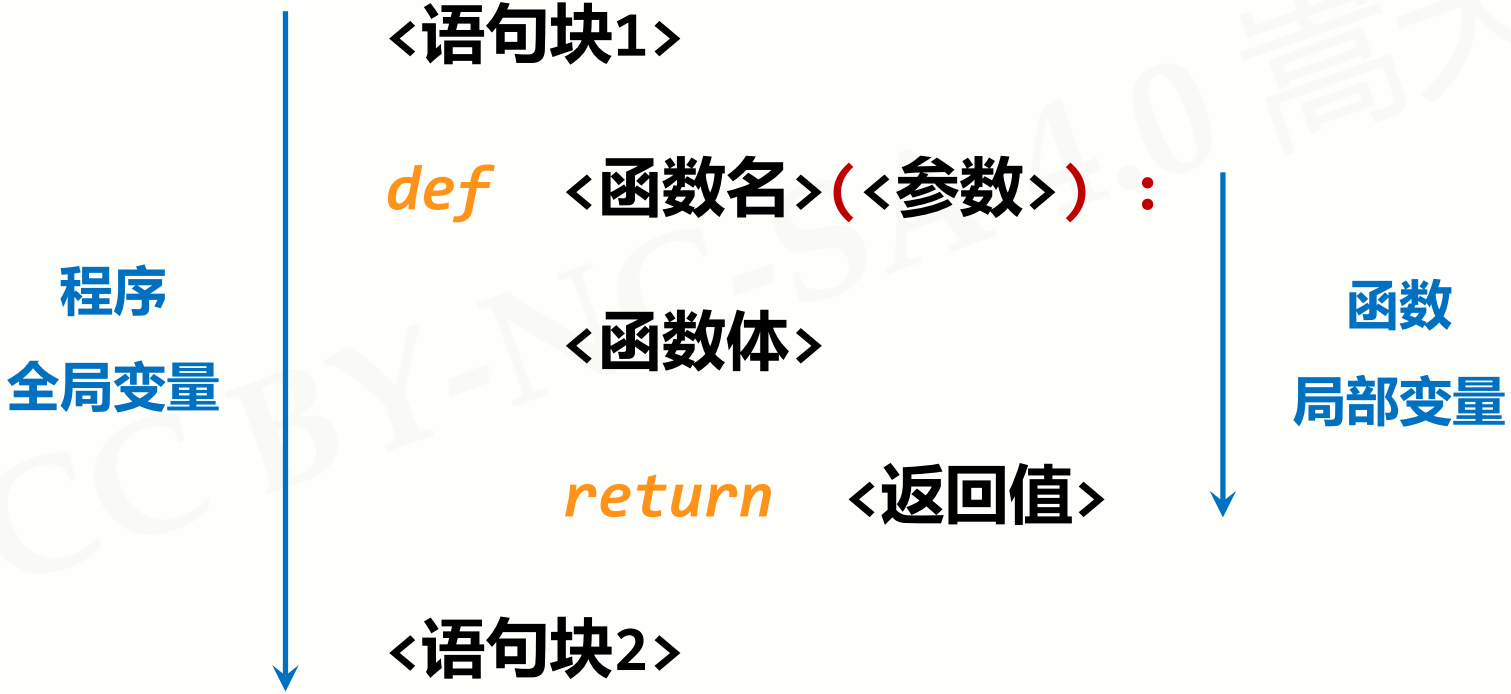
```
>>> fact( 10,5 )
(725760, 10, 5)
>>> a,b,c = fact(10,5)
>>> print(a,b,c)
725760 10 5
```

元组类型



局部变量和全局变量

局部变量和全局变量



局部变量和全局变量

```
n, s = 10, 100
def fact(n) :
    s = 1
    for i in range(1, n+1):
        s *= i
    return s
print(fact(n), s)
```

n和s是全局变量

fact()函数中的n和s是局部变量

n和s是全局变量

运行结果

```
>>>
3628800 100
```

局部变量和全局变量

规则1: 局部变量和全局变量是不同变量

- 局部变量是函数内部的占位符，与全局变量可能重名但不同
- 函数运算结束后，局部变量被释放
- 可以使用*global*保留字在函数内部使用全局变量

局部变量和全局变量

```
n, s = 10, 100
def fact(n) :
    s = 1
    for i in range(1, n+1):
        s *= i
    return s
print(fact(n), s)
```

fact()函数中s是局部变量

与全局变量s不同

此处局部变量s是3628800

此处全局变量s是100

运行结果

```
>>>
3628800 100
```

局部变量和全局变量

```
n, s = 10, 100
def fact(n) :
    global s
    for i in range(1, n+1):
        s *= i
    return s
print(fact(n), s)
```

fact()函数中使用global保留字声明

此处s是全局变量s

此处s指全局变量s

此处全局变量s被函数修改

运行结果

```
>>>
362880000 362880000
```

局部变量和全局变量

规则2: 局部变量为组合数据类型且未创建，等同于全局变量

```
ls = ["F", "f"]
def func(a) :
    ls.append(a)
    return
func("C")
print(ls)
```

通过使用[]真实创建了一个全局变量列表ls

此处ls是列表类型，未真实创建

则等同于全局变量

全局变量ls被修改

运行结果

```
>>>
['F', 'f', 'C']
```

局部变量和全局变量

```
ls = ["F", "f"]
def func(a) :
    ls = []
    ls.append(a)
    return
func("C")
print(ls)
```

通过使用[]真实创建了一个全局变量列表ls

此处ls是列表类型，真实创建

ls是局部变量

局部变量ls被修改

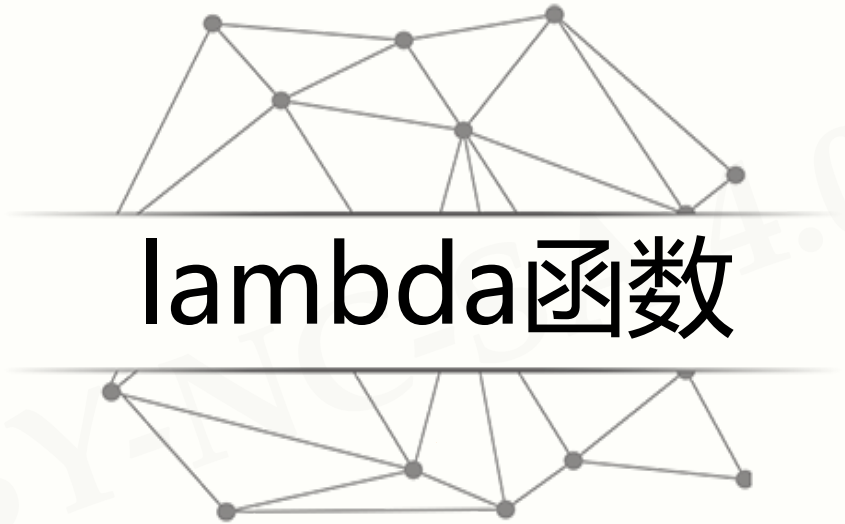
运行结果

```
>>>
['F', 'f']
```

局部变量和全局变量

使用规则

- 基本数据类型，无论是否重名，局部变量与全局变量不同
- 可以通过global保留字在函数内部声明全局变量
- 组合数据类型，如果局部变量未真实创建，则是全局变量



lambda函数

lambda函数

lambda函数返回函数名作为结果

- lambda函数是一种匿名函数，即没有名字的函数
- 使用*Lambda*保留字定义，函数名是返回结果
- lambda函数用于定义简单的、能够在一行内表示的函数

lambda函数

<函数名> = lambda <参数>: <表达式>

等价于

```
def <函数名>(<参数>) :
    <函数体>
    return <返回值>
```

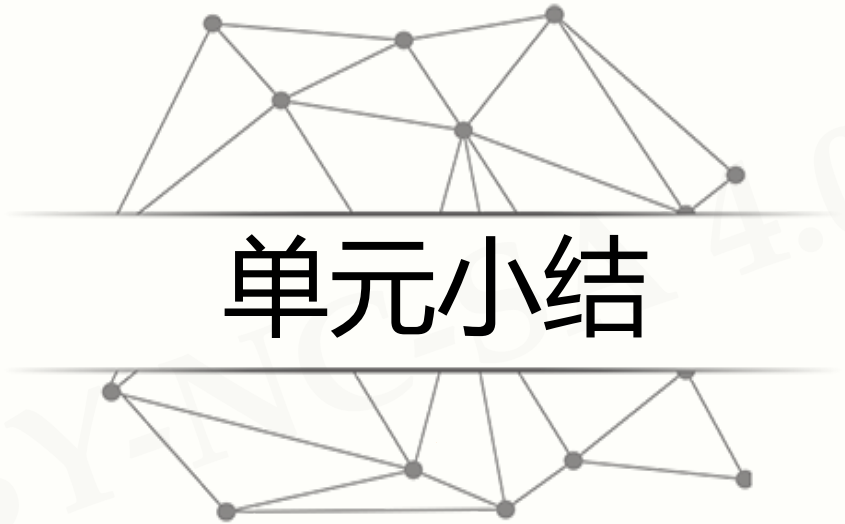
lambda函数

```
>>> f = lambda x, y : x + y
>>> f(10, 15)
25
>>> f = lambda : "lambda函数"
>>> print(f())
lambda函数
```

lambda函数的应用

谨慎使用lambda函数

- lambda函数主要用作一些特定函数或方法的参数
- lambda函数有一些固定使用方式，建议逐步掌握
- 一般情况，建议使用*def*定义的普通函数



单元小结

函数的定义与使用

- 使用保留字*def*定义函数，*lambda*定义匿名函数
- 可选参数(赋初值)、可变参数(*b)、名称传递
- 保留字*return*可以返回任意多个结果
- 保留字*global*声明使用全局变量，一些隐式规则

小花絮

小议"函数式编程"

"函数式编程"用函数将程序组织起来，貌似很流行，为何不早学呢?

- 第一，函数式编程主要源于C语言，Python不是C，这说法不流行
- 第二，不要纠结于名字，关键在于按照"控制流"编程的过程式编程思维
- 第三，Python编程中函数不必须，因此更灵活，更探寻本质

如果您学过其他编程语言，不要被束缚，从本质上看待Python才更有趣!