

python 语言程序设计基础

Hengsheng Zhou

电信与智能制造学院

2025 年 4 月 21 日



郑州西亚斯学院
SIAS UNIVERSITY

Outline

- ① 函数
 - 形参不可变函数的定义
 - 可变形参函数的定义
- ② 函数的调用
- ③ Lambda 表达式
- ④ 正则表达式

1 函数

- 形参不可变函数的定义
- 可变形参函数的定义

2 函数的调用

3 Lambda 表达式

4 正则表达式

1 函数

- 形参不可变函数的定义
- 可变形参函数的定义

2 函数的调用

3 Lambda 表达式

4 正则表达式

definition

函数是一个实现特定功能的代码段

definition

函数是一个实现特定功能的代码段

example

无参函数 `def my_function():`
`print("Hello from a function")` 带有参数的函数 `def my_function(name):`
`print(name + " say hello")` 带有两个参数的函数 `def my_function(name,`
`something):`
`print(name + "say" + something)`

definition

函数是一个实现特定功能的代码段

example

无参函数 `def my_function():`
`print("Hello from a function")` 带有参数的函数 `def my_function(name):`
`print(name + " say hello")` 带有两个参数的函数 `def my_function(name,`
`something):`
`print(name + "say" + something)`

Attention

调用函数时传入的参数数量必须与形参个数相同。

definition

函数是一个实现特定功能的代码段

example

无参函数 `def my_function():`
`print("Hello from a function")` 带有参数的函数 `def my_function(name):`
`print(name + " say hello")` 带有两个参数的函数 `def my_function(name,`
`something):`
`print(name + "say" + something)`

Attention

调用函数时传入的参数数量必须与形参个数相同。

Attention

如果在定义函数时无法确定函数形参数量该如何定义函数？

1 函数

- 形参不可变函数的定义
- 可变形参函数的定义

2 函数的调用

3 Lambda 表达式

4 正则表达式

创建可变形参函数

```
def my_function(*kids):  
    print("The youngest child is " + kids[2])
```

创建可变形参函数

```
def my_function(*kids):  
    print("The youngest child is " + kids[2])
```

调用

```
my_function("Emil", "Tobias", "Linus")# 将实参作为元组传入函数因此  
需要通过索引值的方式访问实参
```

定义

```
def my_function(**kid):  
    print("His last name is " + kid["lname"])
```

定义

```
def my_function(**kid):  
    print("His last name is " + kid["lname"])
```

调用

```
my_function(fname = "Tobias", lname = "Refsnes")# 由于实参以字典  
的方式传入函数, 因此条用时需要指定 key
```

1 函数

2 函数的调用

3 Lambda 表达式

4 正则表达式

规定实参形式

`def test_function (a,b,/,*,c,d,)` # 规定,/符合之前实参按顺序赋值, 在,*之后形参按 key 复制

规定实参形式

`def test_function(a,b,/,*,c,d,)` # 规定,/符合之前实参按顺序赋值,在,*之后形参按 key 复制

为形参设置默认值

`def my_function(country = "Norway")` # 如果调用函数不传入实参怎使用默认值

规定实参形式

```
def test_function (a,b,/,*,c,d,)# 规定,/符合之前实参按顺序赋值, 在,*  
之后形参按 key 复制
```

为形参设置默认值

```
def my_function(country = "Norway")# 如果调用函数不传入实参怎使  
用默认值
```

空体函数

```
def myfunction():# 通过 pass 关键字可以暂时不指定函数体  
pass
```

1 函数

2 函数的调用

3 Lambda 表达式

4 正则表达式

创建

```
x = lambda a, b, c : a + b + c
```

创建

```
x = lambda a, b, c : a + b + c
```

调用

```
def lambda_func(a,b,*,func):
```

定义 `lambda_func(2,4,func=lambda a,b:a*b)`

调用将 `lambda` 表达式作为实参

1 函数

2 函数的调用

3 Lambda 表达式

4 正则表达式

表: 基本表达式

	一个字符序列
位置	^、\$
数量	*, +, ?,
元素	[], ., \
逻辑	

表: 基本表达式

	一个字符序列
位置	\wedge 、 $\$$
数量	$*$ 、 $+$ 、 $?$ 、
元素	$[]$ 、 $.$ 、 \backslash
逻辑	$ $

表: 元素表达式 $[]$

$[abc]$	abc 任意一个
$[a - c]$	字母表任意一个
$[^abc]$	除了 abc 任意一个

表: 基本表达式

	一个字符序列
位置	<code>^</code> 、 <code>\$</code>
数量	<code>*</code> 、 <code>+</code> 、 <code>?</code> 、
元素	<code>[]</code> 、 <code>.</code> 、 <code>\</code>
逻辑	<code> </code>

表: 元素表达式 `[]`

<code>[abc]</code>	abc 任意一个
<code>[a - c]</code>	字母表任意一个
<code>[^abc]</code>	除了 abc 任意一个

表: 元素表达式 `\`

<code>\d</code>	数字
<code>\D</code>	非数字
<code>\s</code>	空格
<code>\S</code>	非空格

example

如何在匹配所有单词