Tedu.cn
达内教育

Python开发进阶

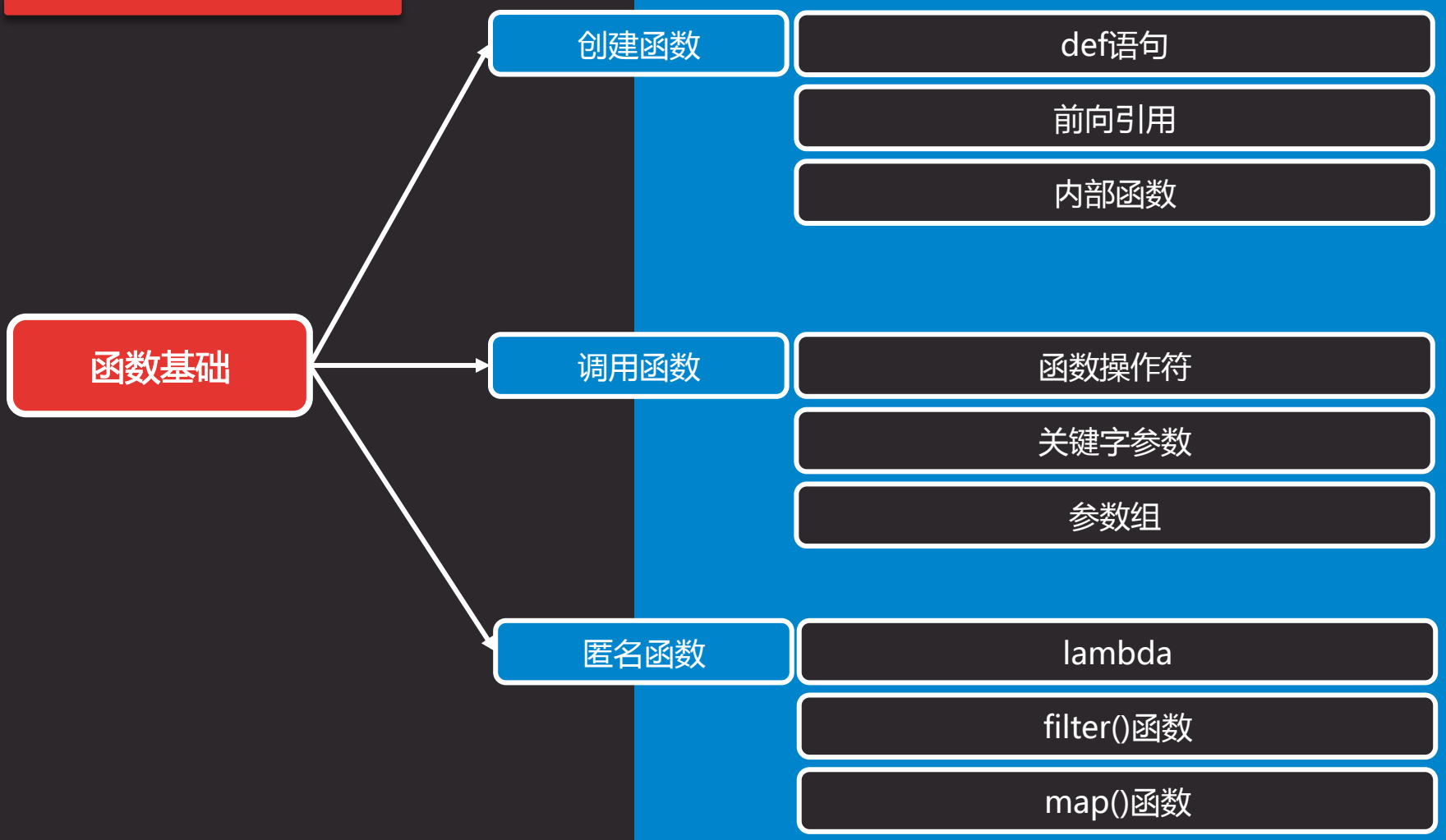
NSD PYTHON2

DAY02

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	函数基础
	10:30 ~ 11:20	
	11:30 ~ 12:00	
下午	14:00 ~ 14:50	函数高级应用
	15:00 ~ 15:50	
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑

函数基础


函数基础
创建函数
def语句
前向引用
内部函数
调用函数
函数操作符
关键字参数
参数组
匿名函数
lambda
filter()函数
map()函数

Tedu.cn
达内教育

创建函数

def 语句

- 函数用def语句创建，语法如下：

```
def function_name(arguments):
    "function_documentation_string"
    function_body_suite
```

知识讲解

- 标题行由def关键字，函数的名字，以及参数的集合（如果有的话）组成
- def子句的剩余部分包括了一个虽然可选但是强烈推荐的文档字串，和必需的函数体

前向引用

- 函数不允许在函数未声明之前对其进行引用或者调用

```
def foo():
    print('in foo')
    bar()

foo()           #报错，因为bar没有定义
--------------------------------------------------------------------
def foo():
    print('in foo')
    bar()

def bar():
    print('in bar’)
foo()           #正常执行，虽然bar的定义在foo定义后面
```

知识讲解

内部函数

- 在函数体内创建另外一个函数是完全合法的，这种函数叫做内部/内嵌函数

```
>>> def foo():
...         def bar():
...             print('bar() is called')
...         print('foo() is called’)
...         bar()
>>> foo()
foo() is called
bar() is called
>>> bar()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'bar' is not defined
```

知识讲解

Tedu.cn
达内教育

调用函数

函数操作符

- 使用一对圆括号()调用函数，如果没有圆括号，只是对函数的引用
- 任何输入的参数都必须放置在括号中

```
>>> def foo():
...     print('Hello world!')
...
>>> foo()
Hello world!
>>> foo
<function foo at 0x7f18ce311b18>
```

知识讲解

关键字参数

- 关键字参数的概念仅仅针对函数的调用
- 这种理念是让调用者通过函数调用中的参数名字来区分参数
- 这样规范允许参数缺失或者不按顺序

```
>>> def get_info(name, age):
...     print("%s's age is %s" % (name, age))
...
>>> get_info(23, 'bob')
23's age is bob
>>> get_info(age = 23, name = 'bob')
bob's age is 23
```

知识讲解

参数组

- python允许程序员执行一个没有显式定义参数的函数
- 相应的方法是通过一个把元组（非关键字参数）或字典（关键字参数）作为参数组传递给函数

```
func(*tuple_grp_nonkw_args, **dict_grp_kw_args)
```

知识讲解

案例1：简单的加减法数学游戏

1. 随机生成两个100以内的数字
2. 随机选择加法或是减法
3. 总是使用大的数字减去小的数字
4. 如果用户答错三次，程序给出正确答案

课堂练习

Tedu.cn
达内教育

匿名函数

lambda

- python允许用lambda关键字创造匿名函数
- 匿名是因为不需要以标准的def方式来声明
- 一个完整的lambda“语句”代表了一个表达式，这个表达式的定义体必须和声明放在同一行

lambda [arg1[, arg2, ... argN]]: expression

```
>>> a = lambda x, y:  x + y
>>> print(a(3, 4))
7
```

知识讲解

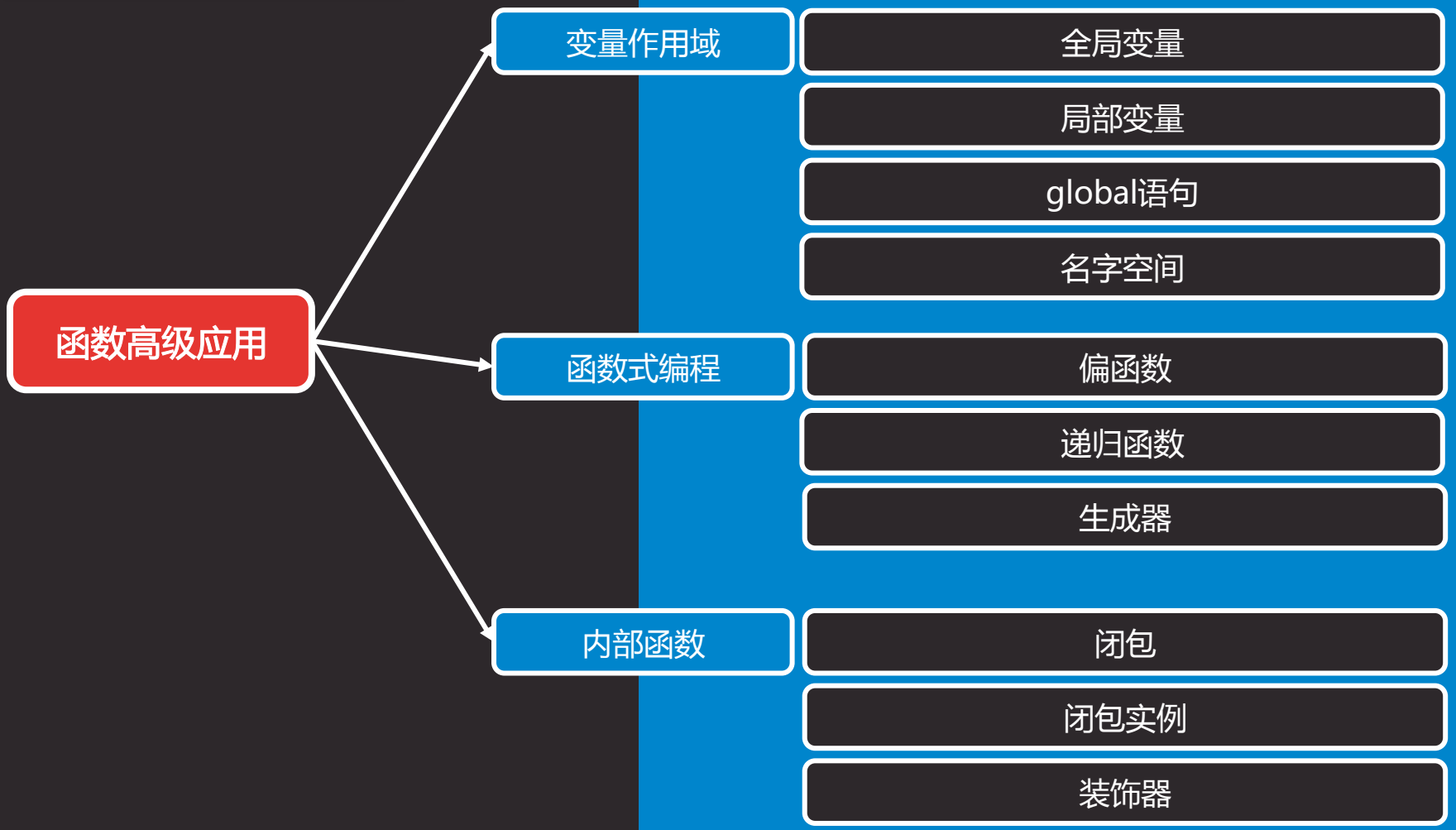
filter()函数

- filter(func, seq)：调用一个布尔函数func来迭代遍历每个序列中的元素；返回一个使func返回值为true的元素的序列
- 如果布尔函数比较简单，直接使用lambda匿名函数就显得非常方便了

```
>>> data = filter(lambda x: x % 2, [num for num in range(10)])
>>> print(data)        #过滤出10以内的奇数
[1, 3, 5, 7, 9]
```

知识讲解

函数高级应用


函数高级应用
变量作用域
全局变量
局部变量
global语句
名字空间
函数式编程
偏函数
递归函数
生成器
内部函数
闭包
闭包实例
装饰器

Tedu.cn
达内教育

变量作用域

全局变量

- 标识符的作用域是定义为其声明在程序里的可应用范围，也就是变量的可见性
- 在一个模块中最高级别的变量有全局作用域
- 全局变量的一个特征是除非被删除掉，否则它们的存活到脚本运行结束，且对于所有的函数，他们的值都是可以被访问的

知识讲解

局部变量

- 局部变量只时暂时地存在，仅仅只依赖于定义它们的函数现阶段是否处于活动
- 当一个函数调用出现时，其局部变量就进入声明它们的作用域。在那一刻，一个新的局部变量名为那个对象创建了
- 一旦函数完成，框架被释放，变量将会离开作用域

知识讲解

局部变量（续1）

- 如果局部与全局有相同名称的变量，那么函数运行时，局部变量的名称将会把全局变量名称遮盖住

```
>>> x = 4
>>> def foo():
...         x = 10
...         print('in foo, x =', x)
...
>>> foo()
in foo, x = 10
>>> print('in main, x =', x)
in main, x = 4
```

知识讲解

global语句

- 因为全局变量的名字能被局部变量给遮盖掉，所以为了明确地引用一个已命名的全局变量，必须使用global语句

```
>>> x = 4
>>> def foo():
...         global x
...         x = 10
...         print('in foo, x =', x)
...
>>> foo()
in foo, x = 10
>>> print('in main, x =', x)
in main, x = 10
```

知识讲解

名字空间

- 任何时候，总有一个到三个活动的作用域（内建、全局和局部）
- 标识符的搜索顺序依次是局部、全局和内建
- 提到名字空间，可以想像是否有这个标识符
- 提到变量作用域，可以想像是否可以“看见”这个标识符

知识讲解

Tedu.cn
达内教育

函数式编程

偏函数

- 偏函数的概念是将函数式编程的概念和默认参数以及可变参数结合在一起
- 一个带有多个参数的函数，如果其中某些参数基本上固定的，那么就可以通过偏函数为这些参数赋默认值

```
>>> from operator import add
>>> from functools import partial
>>> add10 = partial(add, 10)
>>> print(add10(25))
35
```

知识讲解

案例2：简单GUI程序

1. 窗口程序提供三个按钮
2. 其中两个按钮的前景色均为白色，背景色为蓝色
3. 第三个按钮前景色为红色，背景色为红色
4. 按下第三个按钮后，程序退出

课堂练习

递归函数

- 如果函数包含了对其自身的调用，该函数就是递归的
- 在操作系统中，查看某一目录内所有文件、修改权限等都是递归的应用

```
>>> def func(num):
...     if num == 1:
...         return 1
...     else:
...         return num * func(num - 1)
...
>>> print(func(5))
120
>>> print(func(10))
3628800
```

知识讲解

案例3：快速排序

1. 随机生成10个数字
2. 利用递归，实现快速排序

课堂练习

生成器

- 从句法上讲，生成器是一个带yield语句的函数
- 一个函数或者子程序只返回一次，但一个生成器能暂停执行并返回一个中间的结果
- yield 语句返回一个值给调用者并暂停执行
- 当生成器的next()方法被调用的时候，它会准确地从离开地方继续

知识讲解

生成器（续1）

- 与迭代器相似，生成器以另外的方式来运作
- 当到达一个真正的返回或者函数结束没有更多的值返回，StopIteration异常就会被抛出

```
>>> def simp_gen():
...         yield '1'
...         yield '2 -> punch'
>>> mygen = simp_gen()
>>> mygen.next()
'1'
>>> mygen.next()
'2 -> punch'
>>> mygen.next()
... ...
  StopIteration
```

知识讲解

Tedu.cn
达内教育

内部函数

闭包

- 闭包将内部函数自己的代码和作用域以及外部函数的作用结合起来
- 闭包的词法变量不属于全局名字空间域或者局部的--而属于其他的名字空间，带着 “流浪"的作用域
- 闭包对于安装计算，隐藏状态，以及在函数对象和作用域中随意地切换是很有用的
- 闭包也是函数，但是他们能携带一些额外的作用域

知识讲解

闭包实例

- 创建通用的计数器

```
>>> def counter(start=0):
...   count = start
...   def incr():
...     nonlocal count
...     count += 1
...     return count
...   return incr
>>> a = counter()
>>> b = counter(10)
>>> a()
1
>>> b()
11
```

知识讲解

装饰器

- 装饰器是在函数调用之上的修饰
- 这些修饰仅是当声明一个函数或者方法的时候，才会应用的额外调用
- 使用装饰器的情形有：
  - 引入日志
  - 增加计时逻辑来检测性能
  - 给函数加入事务的能力

知识讲解

案例4：测试程序运行效率

1. 有个程序包含多个函数
2. 程序运行耗时较长
3. 为了确定哪个函数是瓶颈，需要计算出每个函数运行时间
4. 要求使用装饰器实现

课堂练习

Tedu.cn
达内教育

总结和答疑