Tedu.cn
达内教育

Python开发入门

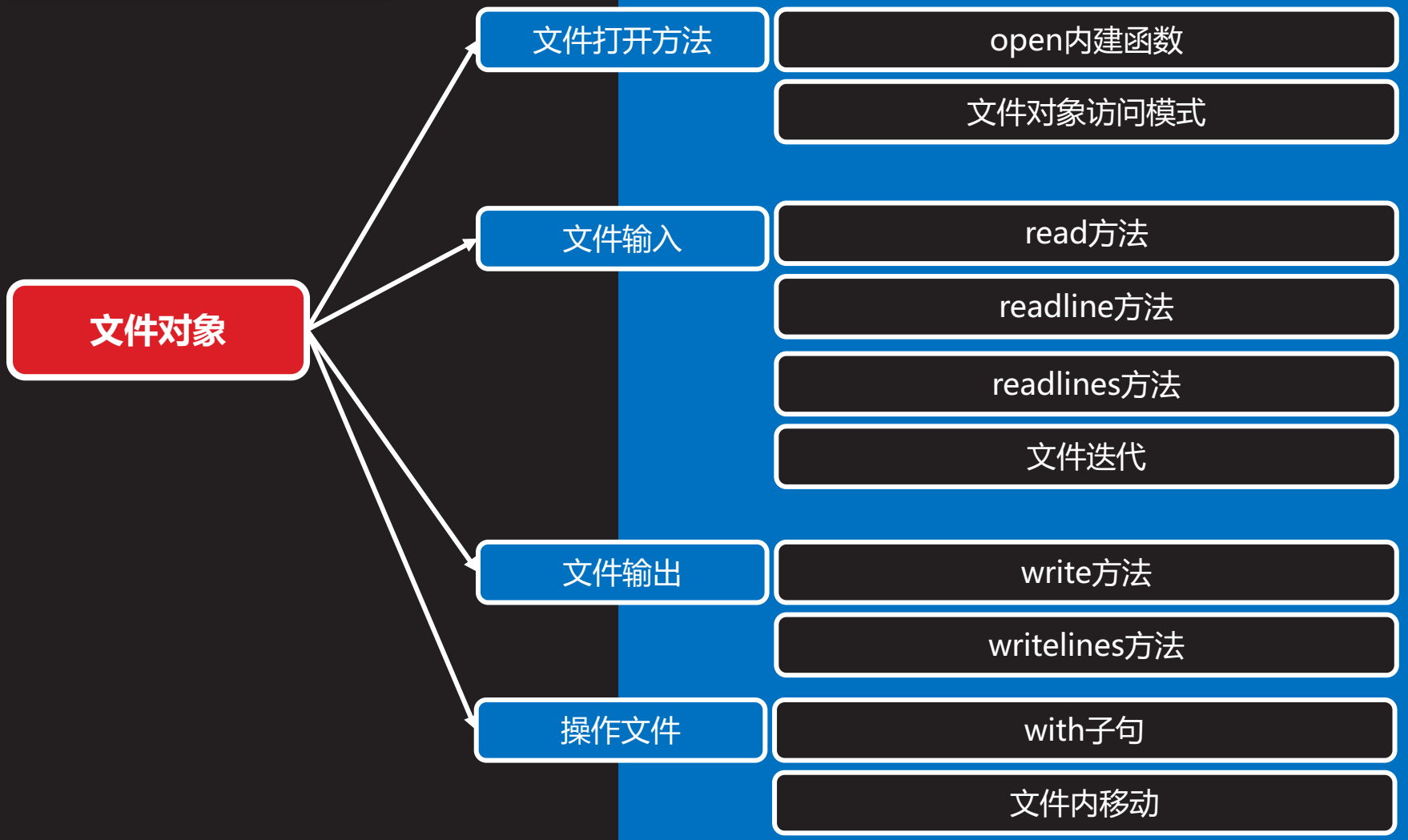
NSD PYTHON1

DAY03

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	文件对象
	10:30 ~ 11:20	
	11:30 ~ 12:00	函数基础
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	模块基础
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑

文件对象


文件对象
文件打开方法
open内建函数
文件对象访问模式
文件输入
read方法
readline方法
readlines方法
文件迭代
文件输出
write方法
writelines方法
操作文件
with子句
文件内移动

Tedu.cn
达内教育

文件打开方法

open内建函数

- 作为打开文件之门的“钥匙”，内建函数open() 提供了初始化输入/输出（I/O）操作的通用接口
- 成功打开文件后时候会返回一个文件对象，否则引发一个错误
- 基本语法：

```
file_object = open(file_name, access_mode='r', buffering=-1)
```

知识讲解

文件对象访问模式

知识讲解

文件模式	操　作
r	以读方式打开（文件不存在则报错）
w	以写方式打开（文件存在则清空，不存在则创建）
a	以追加模式打开（必要时创建新文件）
r+	以读写模式打开（参见r）
w+	以读写模式打开（参见w）
a+	以读写模式打开（参见a）
b	以二进制模式打开

Tedu.cn
达内教育

文件输入

read方法

- read()方法用来直接读取字节到字符串中，最多读取给定数目个字节
- 如果没有给定size参数（默认值为-1）或者size值为负，文件将被读取直至末尾

```
>>> data = fobj.read()
>>> print(data)
```

知识讲解

readline方法

知识讲解

- 读取打开文件的一行（读取下个行结束符之前的所有字节）
- 然后整行，包括行结束符，作为字符串返回
- 它也有一个可选的size参数，默认为-1，代表读至行结束符
- 如果提供了该参数，那么在超过size个字节后会返回不完整的行

```
>>> data = fobj.readline()
>>> print(data)
```

readlines方法

- readlines()方法读取所有（剩余的）行然后把它们作为一个字符串列表返回

```
>>> data = fobj.readlines()
>>> print(data)
```

知识讲解

文件迭代

- 如果需要逐行处理文件，可以结合for循环迭代文件
- 迭代文件的方法与处理其他序列类型的数据类似

```
>>> fobj = open('star.py')
>>> for eachLine in fobj:
...         print(eachLine, end= '')
```

知识讲解

Tedu.cn
达内教育

文件输出

write方法

- write()内建方法功能与read()和readline()相反。它把含有文本数据或二进制数据块的字符串写入到文件中去
- 写入文件时，不会自动添加行结束标志，需要程序员手工输入

```
>>> fobj.write('Hello World!\n')
13
```

知识讲解

writelines方法

- 和readlines()一样，writelines()方法是针对列表的操作
- 它接受一个字符串列表作为参数，将它们写入文件
- 行结束符并不会被自动加入，所以如果需要的话，必须在调用writelines()前给每行结尾加上行结束符

```
>>> fobj.writelines(['Hello World!\n', 'python programing\n'])
```

知识讲解

Tedu.cn
达内教育

操作文件

with子句

- with语句是用来简化代码的
- 在将打开文件的操作放在with语句中，代码块结束后，文件将自动关闭

```
>>> with open('foo.py') as f:
...         data = f.readlines()
...
>>> f.closed
True
```

知识讲解

文件内移动

- seek(offset[, whence])：移动文件指针到不同的位置
  - offset是相对于某个位置的偏移量
  - whence的值，0表示文件开头，1表示当前位置，2表示文件的结尾
- tell()：返回当前文件指针的位置

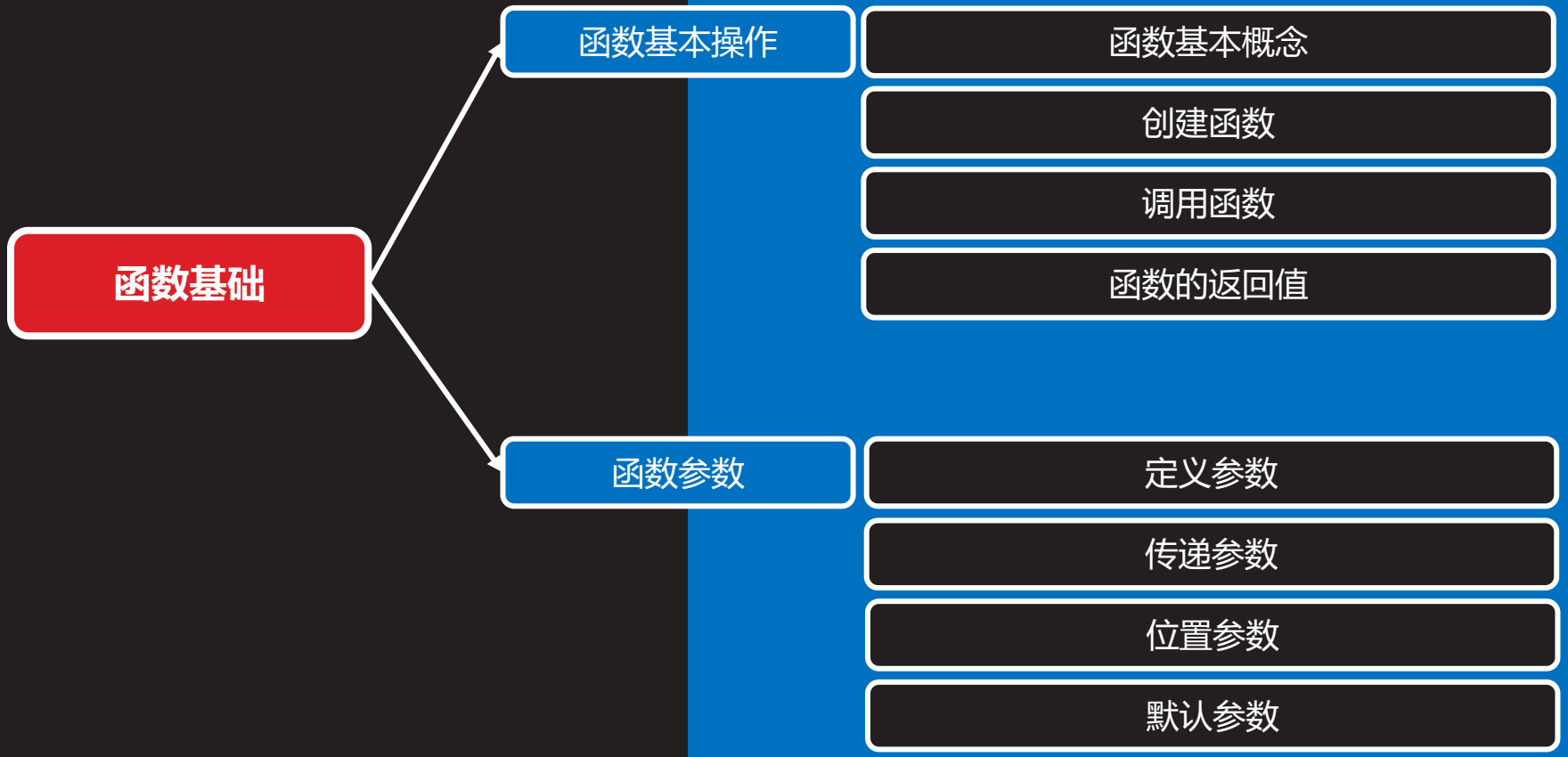
知识讲解

案例1：模拟cp操作

1. 创建cp.py文件
2. 将/bin/ls“拷贝”到/root/目录下
3. 不要修改原始文件

课堂练习

函数基础


函数基础
函数基本操作
函数基本概念
创建函数
调用函数
函数的返回值
函数参数
定义参数
传递参数
位置参数
默认参数

Tedu.cn
达内教育

函数基本操作

函数基本概念

知识讲解

- 函数是对程序逻辑进行结构化或过程化的一种编程方法
- 将整块代码巧妙地隔离成易于管理的小块
- 把重复代码放到函数中而不是进行大量的拷贝，这样既能节省空间，也有助于保持一致性
- 通常函数都是用于实现某一种功能

创建函数

知识讲解

- 函数是用def语句来创建的，语法如下：

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

- 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

调用函数

- 同大多数语言相同，python用一对圆括号调用函数
- 如果没有加圆括号，只是对函数的引用

```
>>> def foo():
...     print('hello')
...
>>> foo()
hello
>>> foo
<function foo at 0x7ff2328967d0>
```

知识讲解

函数的返回值

- 多数情况下，函数并不直接输出数据，而是向调用者返回值
- 函数的返回值使用return关键字
- 没有return的话，函数默认返回None

```
>>> def foo():
...         res = 3 + 4
>>> i = foo()
>>> print i
None
```

知识讲解

Tedu.cn
达内教育

函数参数

定义参数

- 形式参数
  - 函数定义时，紧跟在函数名后（圆括号内）的参数被称为形式参数，简称形参。由于它不是实际存在变量，所以又称虚拟变量
- 实际参数
  - 在主调函数中调用一个函数时，函数名后面括弧中的参数（可以是一个表达式）称为“实际参数”，简称实参

知识讲解

传递参数

- 调用函数时，实参的个数需要与形参个数一致
- 实参将依次传递给形参

```
>>> def foo(x, y):
...         print('x=%d, y=%d' % (x, y))
>>> foo()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: foo() takes exactly 2 arguments (0 given)
>>> foo(3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: foo() takes exactly 2 arguments (1 given)
>>> foo(3, 4)
x=3, y=4
```

知识讲解

位置参数

- 与shell脚本类似，程序名以及参数都以位置参数的方式传递给python程序
- 使用sys模块的argv列表接收

```
[root@zzghost1 day02]# vim args.py
#!/usr/bin/env python3
import sys
print sys.argv

[root@zzghost1 day02]# ./args.py hello world
['./args.py', 'hello', 'world']
```

知识讲解

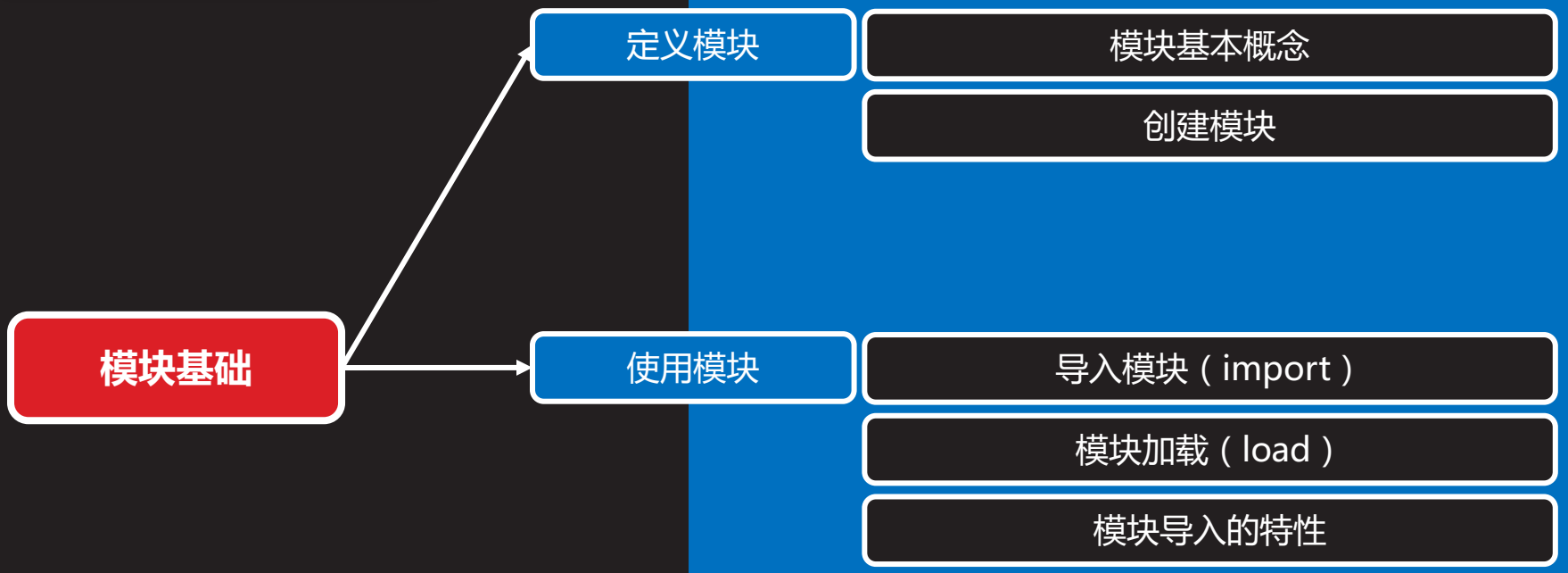
默认参数

- 默认参数就是声明了默认值的参数
- 因为给参数赋予了默认值，所以在函数调用时，不向该参数传入值也是允许的

```
>>> def pstar(num = 30):
...         print('*' * num)
...
>>> pstar()
******************************
>>> pstar(40)
****************************************
```

知识讲解

模块基础



Tedu.cn
达内教育

定义模块

模块基本概念

- 模块是从逻辑上组织python代码的形式
- 当代码量变得相当大的时候，最好把代码分成一些有组织的代码段，前提是保证它们的彼此交互
- 这些代码片段相互间有一定的联系，可能是一个包含数据成员和方法的类，也可能是一组相关但彼此独立的操作函数

知识讲解

创建模块

- 模块物理层面上组织模块的方法是文件，每一个以.py作为结尾的python文件都是一个模块
- 模块名称切记不要与系统中已存在的模块重名
- 模块文件名字去掉后面的扩展名（.py）即为模块名

知识讲解

导入模块（import）

- 使用import导入模块
- 模块属性通过“模块名.属性”的方法调用
- 如果仅需要模块中的某些属性，也可以单独导入

```
>>> import sys
>>> import os, string
>>> string.digits
'0123456789'
>>> from random import randint
>>> randint(1, 10)
3
```

知识讲解

模块加载（load）

- 一个模块只被加载一次，无论它被导入多少次
- 只加载一次可以阻止多重导入时代码被多次执行
- 如果两个文件相互导入，防止了无限的相互加载
- 模块加载时，顶层代码会自动执行，所以只将函数放入模块的顶层是良好的编程习惯

知识讲解

模块导入的特性

- 模块具有一个__name__特殊属性
- 当模块文件直接执行时，__name__的值为'__main__'
- 当模块被另一个文件导入时，__name__的值就是该模块的名字

```
[root@zzghost1 day02]# vim foo.py
#!/usr/bin/env python3
print(__name__)
[root@py01 bin]# ./foo.py
__main__
[root@zzghost1 day02]# python
>>> import foo
foo
```

知识讲解

案例2：生成随机密码

- 创建randpass.py脚本，要求如下：
  1. 编写一个能生成8位随机密码的程序
  2. 使用random的choice函数随机取出字符
  3. 改进程序，用户可以自己决定生成多少位的密码

课堂练习

Tedu.cn
达内教育

总结和答疑