# 第1章 开启Python学习之旅

1. **填空题**
2. 对象
3. 可移植性
4. **判断题**
5. √
6. ×
7. ×
8. **选择题**
9. D
10. B
11. D
12. B
13. D
14. **简答题**
15. 答：简单易学，免费开源，可移植性、面向对象、丰富的库。
16. 答：
17. Python2使用print语句进行输出，Python3使用print()函数进行输出。
18. Python 2中字符串有默认ASCII编码的str类型和unicode类型。Python 3中默认使用UTF-8编码，以更好地实现对中文或其它非英文字符的支持。
19. 使用运算符“/”进行除法运算时，Python 2中整数相除的结果是一个整数，浮点数相除的结果是一个浮点数；Python 3中整数相除的结果也会得到浮点数。
20. 在Python 2中，所有类型的对象直接被抛出，在Python 3中，只有继承自BaseException的对象才可以被抛出。在Python 2中，捕获异常的语法是“except Exception, err”；在Python 3中，捕获异常的语法变更为“except Exception as err”。 在Python 2中，处理异常可以使用“raise Exception, args”或者“raise Exception(args)”两种语法；在Python 3中，处理异常只能使用“raise Exception(args)”。 Python 3取消了异常类的序列行为和message属性。

# 第2章 数字类型与字符串

1. **填空题**
2. 复数类型，布尔类型
3. 整型
4. 实部，虚部
5. **判断题**
6. √
7. ×
8. ×
9. ×
10. **选择题**
11. D
12. C
13. A
14. BD
15. D
16. **编程题**
17. 答案：

print("I'm from China")

1. 答案：

string = ' python是一种解释型语言 '

string = string.strip().capitalize()

print(string)

1. 答案：

string = 'tsacti'

print(string[::-1])

# 第3章 流程控制

1. **填空题**
2. for，while
3. if
4. True
5. **判断题**
6. ×
7. ×
8. √
9. ×
10. **选择题**
11. B
12. A
13. C
14. A
15. B
16. **编程题**
17. 答案：

res = 0

for i in range(101):

res = res + i

print(res)

1. 答案：

i = 0

while i < 100:

if i % 2 == 0:

print(i)

i += 1

1. 答案：

i = 2

for i in range(2, 100):

j = 2

for j in range(2, i):

if(i % j == 0):

break

else:

print(i)

# 第4章 列表与元组

1. **填空题**
2. list
3. 切片
4. tuple
5. **判断题**
6. ×
7. ×
8. ×
9. √
10. **选择题**
11. A
12. A
13. C
14. A
15. C
16. **编程题**

li\_num1 = [4, 5, 2, 7]

li\_num2 = [3, 6]

li\_num1.extend(li\_num2)

li\_num1.sort(reverse=True)

print(li\_num1)

tu\_num1 = ('p', 'y', 't', ['o', 'n'])

tu\_num1[3].append('h')

print(tu\_num1)

# 第5章 字典和集合

1. **填空题**
2. 键 值
3. 唯一
4. keys()
5. 元素
6. 无序 元素不可重复
7. **判断题**
8. √
9. √
10. ×
11. √
12. **选择题**
13. A
14. B
15. B
16. C
17. B
18. **编程题**

# 遍历字符串，使用字典记录各个字符及其出现的次数

str = 'skdaskerkjsalkj'

count = {}

for i in str:

if i in count.keys():

count[i] += 1

else:

count[i] = 1

print(count)

li\_one = [1,2,1,2,3,5,4,3,5,7,4,7,8]

li\_one = list(set(li\_one))

print(li\_one)

# 第6章 函数

1. **填空题**
2. def
3. lambda
4. global
5. **判断题**
6. √
7. √
8. √
9. ×
10. √
11. **选择题**
12. D
13. C
14. B
15. C
16. B
17. **简答题**
18. 匿名函数是无需函数名标识的函数，它的函数体只能是单个表达式。Python中使用关键字lambda定义匿名函数，可实现的功能比较单一；不能被其它程序使用。
19. 位置参数：实际参数按照位置顺序依次传递给形式参数。

关键字参数：通过“形式参数=实际参数”的格式将实际参数与形式参数相关联，根据形参的名称进行参数传递。

不定长参数：不定长参数分为\*args和\*\*kwargs两种，它们都接收不定数量的位置参数，其中\*args接收的多个参数以元组形式保存，\*\*kwargs接收的多个参数以字典形式保存。

1. **编程题**

sum = 0

for i in range(0,101):

if i % 2 == 0:

sum += i

print(sum)

def factorial(num):

if num == 2:

return 1

else:

return num \* factorial(num - 1)

if \_\_name\_\_ == '\_\_main\_\_':

print(factorial(20))

# 第7章 面向对象程序设计

1. **填空题**
2. class
3. self
4. super()
5. 双下画线
6. **判断题**
7. ×
8. √
9. √
10. √
11. **选择题**
12. D。解析：实例无法修改类属性，但可以访问类属性。
13. A
14. B
15. A
16. C
17. **简答题**
18. 构造方法：类的默认可选方法，如果定义类时不显式提供，Python会给出默认的\_\_init\_\_()方法；构造方法在类实例化时自动调用。

析构方法：类的默认可选方法，如果定义类时不显式提供，Python会给出默认的\_\_del\_\_()方法；构造方法在对象被清理时自动调用。

1. 类方法的第一个参数为默认参数cls，静态方法没有默认参数；类方法用关键字@classmethod修饰，静态方法用关键字@staticmethod修饰。
2. Python支持继承机制，在定义类时可以让类继承已有类，如此新定义的类可直接获取已有类中的成员；Python中的继承分为单继承和多继承，被继承的类称为父类或基类，新定义的类称为子类或派生类，继承一般用于扩展类的功能；子类可以重写从父类继承来的方法，亦可定义新的方法。
3. **编程题**

答案：

class Circle:

radius = 1

def \_\_init\_\_(self,r):

self.radius = r

def get\_perimeter(self):

perimeter = 2\*3.14\*self.radius

return perimeter

def get\_area(self):

area = 3.14 \* self.radius \* self.radius

return area

circle = Circle(4)

per = circle.get\_perimeter()

area = circle.get\_area()

print('周长为：%.2f'%per)

print('面积为：%.2f'%area)

# 第8章 模块

1. **填空题**
2. 第三方模块 自定义模块
3. import 模块 from 模块 import 函数/方法
4. \_\_init\_\_.py
5. **判断题**
6. √
7. √
8. ×
9. ×
10. ×
11. **选择题**
12. D
13. A
14. D
15. B
16. D
17. **简答题**
18. 编简述包中\_\_init\_\_.py文件的作用。

答：\_init\_\_.py文件有两个作用，第一个作用是标识当前目录是一个Python的包；第二个作用是模糊导入，如果\_\_init\_\_文件中没有声明\_\_all\_\_属性，那么使用from ... import \*导入的内容为空 。

1. 请简述\_\_name\_\_属性的用法。

答：\_\_name\_\_这个系统变量显示了当前模块执行过程中的名称，如果当前程序运行在这个模块中，\_\_name\_\_ 的名称就是\_\_main\_\_如果不是，则为这个模块的名称。\_\_main\_\_一般作为函数的入口，常常有if \_\_name\_\_ == "\_\_main\_\_":来表明整个工程开始运行的入口。

# 第9章 文件与文件路径操作

1. **填空题**
2. close()
3. join()
4. listdir()
5. **判断题**
6. ×
7. √
8. √
9. √
10. √
11. **选择题**
12. C。readlines()方法返回值是一个列表。
13. C
14. C
15. C
16. A
17. **简答题**
18. 文件相对路径指这个文件夹所在的路径与其它文件（或文件夹）的路径关系，绝对路径指盘符开始到当前位置的路径。
19. 文件读写位置属性用于记录文件当前读写的位置。
20. **编程题**

答案：

source\_file = open('words\_file.txt', 'r',encoding='utf-8')

new\_file = open('new\_file.txt','w+',encoding='utf-8')

content = source\_file.read(1)

while content:

if content == 'Z':

content = 'A'

elif content == 'z':

content = 'a'

elif (ord(content) in range(65,90)) or (ord(content) in range(97,122)):

content = chr(ord(content)+1)

new\_file.write(content)

content = source\_file.read(1)

source\_file.close()

new\_file.close()

# 第10章 错误和异常

1. **填空题**
2. BaseException
3. try
4. 上下文管理器
5. **判断题**
6. √
7. √
8. √
9. √
10. **选择题**
11. D
12. B
13. A
14. A
15. A
16. **简答题**
17. 简述try-except的用法和作用。

答：作用：try-except语句用于捕获程序运行时的异常；用法：try子句后面是可能出错的代码，except子句后面是捕获的异常类型。

1. with语句如何实现资源的自动关闭。

答：with语句适用于对资源进行访问的场合，无论资源在使用过程中是否发生异常，都可以使用with语句保证执行释放资源操作，这是因为with语句使用了上下文管理的技术管理资源，上下文管理技术中提供\_\_enter\_\_()和\_\_exit\_\_()方法，进入上下文管理器时调用\_\_enter\_\_()方法，当离开上下文管理器时调用\_\_exit\_\_()方法。

# 第11章 正则表达式

1. **填空题**
2. re
3. sub() subn()
4. 非贪婪
5. **判断题**
6. √
7. √
8. √
9. ×
10. ×
11. **选择题**
12. D
13. D
14. D
15. A
16. A
17. **简答题**

import re

str\_data = ['https://www.baidu.com','www.baidu.com',

'https://www.qq.com/',]

url\_regex = re.compile('^http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|'

'[!\*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')

for url in str\_data:

print(url\_regex.findall(url))

import re

str\_data = ['15732xx@163.com','123@qq.com','483qq.com']

email\_regex = re.compile('^[0-9a-zA-Z\_]{0,19}@[0-9a-zA-Z]'

'{1,13}\.[com,cn,net]{1,3}$')

for email in str\_data:

print(email\_regex.findall(email))

# 第12章 图形用户界面编程

1. **填空题**
2. messagebox
3. text
4. Entry()方法
5. Menu()方法
6. **判断题**
7. ×
8. √
9. √
10. √
11. √
12. **选择题**
13. A
14. B
15. A
16. A
17. A

# 第13章 进程和线程

1. **填空题**
2. 进程
3. 0
4. 共享
5. 前台
6. 锁定
7. **判断题**
8. √
9. ×
10. √
11. ×
12. √
13. **选择题**
14. C
15. B
16. D
17. A
18. D
19. **简答题**
20. 答：主线程的作用主要有两个，一个是产生其他子线程，另一个是最后执行各种关闭操作。
21. 答：死锁是指两个或两个以上的线程在执行过程中，由于各自持有一部分共有资源或者彼此通信而造成的一种阻塞的现象。
22. **编程题**

import threading

import time

class PrintNum(threading.Thread):

def run(self):

for i in range(1, 52, 2):

lock\_show2.acquire()

print(i, end='')

print(i + 1, end='')

time.sleep(0.2)

lock\_show1.release()

class PrintWord(threading.Thread):

def run(self):

for i in range(26):

lock\_show1.acquire()

print(chr(i + ord('A')))

time.sleep(0.2)

lock\_show2.release()

lock\_show1 = threading.Lock()

lock\_show2 = threading.Lock()

thread\_one = PrintNum()

thread\_two = PrintWord()

# 因为线程执行顺序是无序的，保证show1()先执行

lock\_show1.acquire()

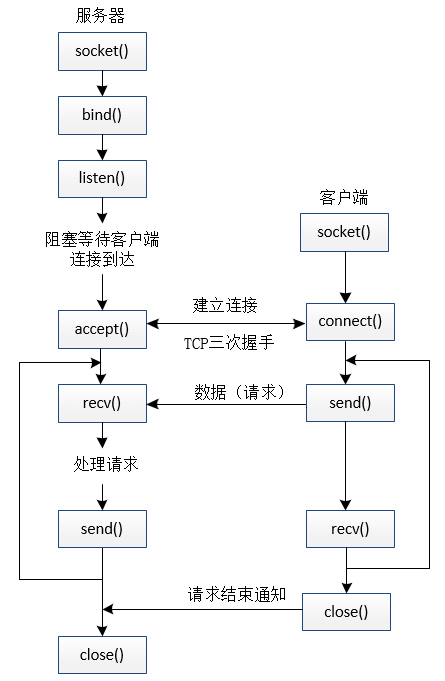
thread\_one.start()

thread\_two.start()

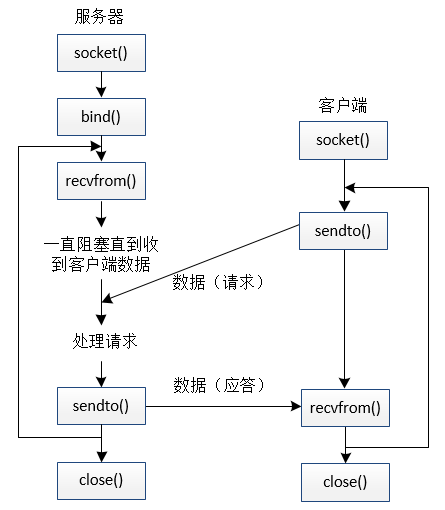
# 第14章 网络编程

1. **填空题**
2. 传输
3. 0~65535
4. socket()
5. 27-2,214-2，221-2
6. recv() recvfrom()
7. **判断题**
8. √
9. √
10. ×
11. ×。在python3中，所有数据的传输必须用bytes类型（bytes只支持ascii码），所以在发送数据时需在发送的字符串前面加 'b'，或使用encode('utf-8')将字符串转换成bytes类型并在接收端使用decode()进行转码。
12. √
13. **选择题**
14. C
15. D
16. B
17. D
18. B
19. **简答题**

基于TCP协议的通信流程：



基于UDP协议的通信流程：



1. **编程题**

服务器端：

#01\_tcp\_ser.py

import socket

def main():

#1.create socket

tcp\_ser\_sock = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)

#2.bind

tcp\_ser\_sock.bind(('192.168.40.129',8080))

#3.listen

tcp\_ser\_sock.listen(1000)

while True:

#4.accept

print("waiting for client...")

cli\_sock,cli\_addr = tcp\_ser\_sock.accept()

#print user info

print("a guest arrived(%s)" % str(cli\_addr))

while True:

#5.recv msg

recv\_data = cli\_sock.recv(1024)

print("recvived msg:%s" % recv\_data.decode("utf-8"))

if recv\_data:

result = eval(recv\_data.decode("utf-8"))

cli\_sock.send(str(result).encode("utf-8"))

else:

break

#6.after communication,close cli\_sock

cli\_sock.close()

print("sign off.")

tcp\_ser\_sock.close()

if \_\_name\_\_=="\_\_main\_\_":

main()

客户端：

#01\_tcp\_cli.py

import socket

#tcp client process

def main():

#1.create sock

cli\_sock = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)

#2.connect server

cli\_sock.connect(("192.168.40.129",8080))

#3.send msg

while True:

send\_msg = input("pls input sth:")

cli\_sock.send(send\_msg.encode('utf-8'))

recvInfo = cli\_sock.recv(1024).decode('utf-8')

print('%s'%recvInfo)

#4.close socket

cli\_sock.colse()

if \_\_name\_\_=="\_\_main\_\_":

main()

# 第15章 数据库编程

1. **填空题**
2. 数据结构
3. 关系模型
4. 文档结构
5. 连接
6. **判断题**
7. √
8. ×
9. √
10. ×
11. **选择题**
12. C
13. ABCD
14. A
15. B
16. D
17. **简答题**
18. 答：数据库是按照数据结构来组织、存储和管理数据的仓库，它可以被看作电子化的文件柜——存储文件的处所，用户可以对文件中的数据进行增加、删除、修改、查找等操作。
19. 关系型数据库是指采用关系模型（即二维表格形式）组织数据的数据库系统，它由数据表和数据表之间的关系组成；非关系型数据库也被称为NoSQL（Not Only SQL）数据库，是指非关系型的、分布式的数据存储系统。与关系型数据库相比，非关系型数据库无需事先为要存储的数据建立字段，它没有固定的结构，既可以拥有不同的字段，也可以存储各种格式的数据。
20. **编程题**

import pymysql

# 连接数据库

conn = pymysql.connect(

host='localhost',

user='root',

password='123456',

charset='utf8'

)

# 获得游标

cursor = conn.cursor()

# 创建数据库

db\_student = "create database if not exists db\_student"

cursor.execute(db\_student)

# 选择数据库

sql\_use = 'use db\_student'

cursor.execute(sql\_use)

# 创建数据库

sql\_table = 'create table if not exists students(stuID int primary key, ' \

'stuName varchar(20), stuGender varchar(20), stuAge int)'

cursor.execute(sql\_table)

# 插入数据

sql\_one = "insert into students (stuID, stuName, " \

"stuGender, stuAge) values (%d, '%s', '%s', %d)"

data1 = (1, '张三', '女', 20)

data2 = (2, "李四", '男', 21)

data3 = (3, "王五", '女', 20)

data4 = (4, "赵六", '男', 19)

data5 = (5, "孙七", '女', 22)

for i in [data1, data2, data3, data4, data5]:

cursor.execute(sql\_one % i)

conn.commit()

# 修改数据

sql = "update students set stuAge = '%d' where stuID = %d"

data = (21, 5)

cursor.execute(sql % data)

conn.commit()

# 查询数据

sql = "select stuID, stuName from students where stuGender = '女'"

cursor.execute(sql)

for row in cursor.fetchall():

print("学号：%d 姓名：'%s'" % row)

# 删除数据

sql = "delete from students where stuID = %d limit %d"

data = (5, 1)

cursor.execute(sql % data)

conn.commit()

print('共删除%d条数据' % cursor.rowcount)

cursor.close() # 关闭游标

conn.close() # 关闭连接

# 第16章 Django框架介绍

1. **填空题**
2. 静态文件
3. WSGI
4. django-admin startproject 项目名称
5. python manage.py startapp 应用名称
6. python manage.py runserver
7. **判断题**
8. ×（URL由协议名、服务器地址、资源路径组成）
9. √
10. √
11. √
12. √
13. **选择题**
14. C
15. A
16. C
17. D
18. B
19. **简答题**

1.

答：WSGI，全称 Web Server Gateway Interface，或者 Python Web Server Gateway Interface ，是为 Python 语言定义的 Web 服务器和 Web 应用程序或框架之间的一种简单而通用的接口。

2.

答：（1）服务器接收浏览器发来的请求后交由View视图进行处理；

（2）View视图接收数据后，将数据传递Model模型中；

（3）Model模型将数据保存到数据库中；

（4）数据库将保存结果返回给Model模型；

（5）Model模型将保存结果返回给View视图；

（6）View视图将结果传递到Template模板；

（7）Template模板根据View视图传递的结果生成HTML页面并返回给View视图；

（8）View视图将HTML页面交由浏览器进行解析展示