1. 程序结构：

从上到下逐行执行

1. 文法规则：

2.1源文件编码

ACSII 编码，区分大小写。中文字符是未定义的.

2.2关键字

None True False def return break continue if elif else while or and not

关键字不可作为变量名或函数名

* 1. 空白字符的处理

空格、制表符在源文件中可以区分词素 (Token)，同时在一行的开头可以表示缩进 。

换行符表示新的语句的开始。

* 1. 注释

从# 开始到本行结束的内容都会被作为注释。

* 1. 标识符

标识符的第一个字符必须是英文字母，第二个 字符开始可以是英文字母、数字或者下划线。标识符区分大小写。长度超过64个字符的标识符是未定义的.

* 1. 常量
     1. 逻辑常量

True为真，False为假

* + 1. 整数常量

整数常量以十进制表示。整数常量不设负数，负数可以由正数取负号得到。

整数的范围是没有限制的，这意味着你必须实现高精度整数。首位为 0 的整数常量是未定义的。

* + 1. 字符串常量

字符串常量是由双引号或单引号括起来的字符串。

可以由两个字符串拼接而形成新的字符串常量。如“123”“456”相当于“123456”

字符串中的所有字符必须是可示字符（printable character），空格或者转义字符中的一种。转义字符与c++中的转义字符相同。

* + 1. 空值常量

None用来表示变量没有指向任何值。

1. 运算符

3.1算术运算符

+,-,\*,/

3.2关系运算符

<,>,<=,>=,==,!=

连续比较：若出现1<2>3这样连续的关系运算符，将它拆成相邻的比较并用and连接，如(1<2) and (2<3)

* 1. 逻辑运算符

and or not

and的定义如下：先计算运算符左边的值，转为bool类型，若为false，返回false，否则计算运算符右边的值，转为bool类型，并返回它。

or ，not的定义可类比

注意短路求值

* 1. 赋值运算符

=

给一个变量赋值的意义是将这个变量指向右值，右值不管类型。

对于一个之前未定义的变量，赋值运算会先定义变量。

全局变量的生效范围是全部范围，局部变量的生效范围是在当前语句块（被缩进和取消缩进包起来的部分），具体局部变量和全局变量划分规则和c++一样。

如a=1,a=”123”,a=1.1这3条语句依次执行，再输出a，结果是1.1

可以连等，如a=b=”123”,意思是b=”123”,a=b这两条语句依次执行

可以给多变量赋值，如a,b=1,2意思是a=1,b=2依次执行

* 1. 增量赋值

+=，-=，\*=，/=

对字符串而言+=就是往后加字符，\*=就是把字符串复制多边加起来，剩下两个符号无定义

* 1. 圆括号

()

圆括号除了用在表达式里，可以用来调用函数，定义函数。

* 1. 优先级

() > or > and > not > comparison > +,- > \*,/

1. 数据类型

bool 只有True False两个值

int 高精度整数

float c++中的double

str 字符串，immutable

1. 语句

5.1 变量定义/赋值语句

变量名=(变量名=…变量名=)值

语法规则参见3.4

5.2 表达式语句

变量名+=值

运算时如果两个运算数类型不一样，按照c++规则执行自动类型转换

* 1. 条件语句

if expression1:

语句块

elif expression2:

语句块

else expression3:

语句块

elif(相当于else if) , else可以没有

* 1. 循环语句

while expression :

语句块

* 1. 跳转语句

break ,return ,continue

1. 函数

6.1 函数定义

def func\_name (parameters):

语句块

参数列表如a,b,c，变量名之间用逗号分隔,可以为空。有些变量可以有默认值，但都必须出现在无默认值的变量后面，如(a,b=1)合法，但(a=1,b)不合法

6.2 函数调用

函数名（传入参数列表） 如foo(1,2)

函数调用必须出现在该函数定义之后。

* 1. 内建函数

print 输出

int 将float或bool转成int

float 将int或bool转成float

str 将int或float或bool转成int

都只有一个参数

1. 其他

如果有不懂的文法规则参考python3.g4，以上面的规则为准

其他如果有问题的可以及时联系助教明确语法。