**17Python\_01Python概述+环境准备+Python起步+数据类型概述**

**一 Python特点**

**高级**: 有高级的数据结构,缩短开发时间与代码量

**面向对象**: 为 数据和逻辑相分离的结构化和过程化编程 添加了新的活力

**可升级:** 提供了基本的开发模块,可以在它上面开发软件,实现代码的重用

**可扩展:** 通过将其分离为多个文件或模块加以组织管理

**可移植性:** Python是用C写的,由于C的可移植性,使Python可以运行在任何

带有ANSI C编译器的平台上

**易学:** 没有其他语言通常用来访问变量\定义代码块和进行模式匹配的命令式符号

**内存管理器:** 内存管理是由Python解释器负责的

**二 环境准备**

**2.1 获取Python3源码**

官方站点: <http://www.python.org>

选择正确的系统

选择正确的版本

**2.2 安装Python3**

**安装依赖包**

yum -y install gcc gc-c++ zlib-devel openssl-devel readline-devel libffi-devel sqlite-devel tcl-devel tk-devel

安装Python3

room9pc01 /]# tar -Jxf Python-3.8.0.tar.xz

room9pc01 /]# cd Python-3.8.0/

room9pc01 Python-3.8.0]# ./configure --prefix=/usr/local

room9pc01 Python-3.8.0]# make && make install

**2.3 设置环境变量**

Python虚拟环境只是一个隔离的空间,对应一个文件夹

配置好虚拟环境后,安装软件包,可以安装到虚拟环境

如果项目完成了,虚拟环境目录可以直接删除

**创建虚拟环境**

room9pc01 ~]# python3 -m venv ~/nsd1906

[root@room9pc01 ~]# ls ~/nsd1906

bin include lib lib64 pyvenv.cfg

**激活虚拟环境**

room9pc01 ~]# source ~/nsd1906/bin/activate

(nsd1906) [root@room9pc01 ~]#

**在虚拟环境下,使用Python就是使用Python3**

(nsd1906) [root@room9pc01 ~]# python --version

Python 3.8.0

(nsd1906) [root@room9pc01 ~]# which python

/root/nsd1906/bin/python

**2.4 设置pycharm**

Pycharm是由JetBrains打造的一款Python IDE

支持的功能有:

调试、语法高亮

Project管理、代码跳转

智能提示、自动完成

单元测试、版本控制

• 下载地址:

https://www.jetbrains.com/pycharm/download

• 分为收费的专业版和免费的社区版

**下载并安装Pycharm社区版**

网址：https://www.jetbrains.com/pycharm/download

1)释放文件：

localhost ~]# tar -xzf pycharm-community-2018.1.1.tar.gz

2)运行下面的命令进入PyCharm 目录：

localhost pycharm-community-2018.1.1]# cd pycharm-community-2018.1.1/bin

3)通过运行下面的命令来运行PyCharm进入图形化安装界面：

localhost bin]# sh pycharm.sh &

4)Pycharm打开后，如果你需要导入之前安装版本的设置的话，可以选择第一个选项，如果没有的话，选择(Do not import settings)默认不导入设置，点击OK，就可以进入pycharm进行配置.

5)激活Pycharm：在弹出的激活窗口中，选择“License serveer”输入激活服务器地址“http://127.0.0.1:1017”，之后点击‘Activate’，完成pycharm激活.

6)启动完成进入欢迎界面，如图-5所示：



**三 Python起步-Python运行方式**

**3.1 交互解释器**

命令行输入命令: python3; ctrl+d 退出

**3.2 文件形式**

3.2.1 明确指定解释器

room9pc01 ~]# python3 hello.py

3.2.2 赋予Python文件可执行权限

room9pc01 ~]# chmod +x hello.py

room9pc01 ~]# ./hello.py

**四 Python语法结构**

**4.1 语句块缩进**

python代码块通过缩进对齐表达代码逻辑而不是使用大括号

缩进表达一个语句属于哪个代码块

缩进风格: **4个空格:非常流行, 范·罗萨姆支持的风格(pycharm中可使用TAB)**

**缩进相同的一组语句构成一个代码块,称之代码组**

首行以关键字开始,以冒号 **:** 结束,该行之后的一行或多行代码构成代码组

如果代码组只有一行,可以将其直接写在冒号后面,但是这样的写法可读性差,不推荐

**4.2 注释及续行**

首要说明的是: 尽管Python是可读性最好的语言之一,这并不意味着程序员在代码中就可以不写注释

和很多UNIX脚本类似,python**注释语句从#字符开始**

注释可以在一行的任何地方开始,解释器会忽略掉该行#之后的所有内容

选中pycharm中的一些语句,Ctrl+/(和?号同一键)注释语句,再按或Ctrl+z取消

一行过长的语句可以使用**反斜杠\分解**成几行

**4.3 同行多个语句**

• 分号 ; 允许你将多个语句写在同一行上

• 但是些语句不能在这行开始一个新的代码块

• 因为可读会变差,所以**不推荐使用**

**4.4 输出语句**

**字符串必须用引号引起来,单双引号无区别**

**输出多项变量时,用逗号分隔变量,输出时各项间默认以空格分隔**

**sep=’\*\*\*’ 可设置输出的各项间以\*\*\*为分隔符**

**字符串和数字不能拼接,报错**

**字符串拼接使用+号,各变量的字符串输出结果之间无分隔符**

• 获取帮助

>>> help(print)

• 使用方式

>>> print('Hello World!') Hello World!

>>> print('Hello' + 'World!') HelloWorld!

>>> print('Hello','World!') Hello World!

>>> print('Hello','World!',sep='\*\*\*') Hello\*\*\*World!

>>> print('Hello','World!',sep='\*\*\*',end='') Hello\*\*\*World!

**4.5 输入语句**

• 获得帮助

>>> help(input)

• 使用方式(注意,**返回值一定是字符类型**)

>>> num = input("Number: ")

Number: 20

>>> num + 10

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: **must be str**, not int

案例:

user = input('请输入用户名:') #以下三种方式输出结构及结果一样

print('欢迎您: ' + user) #注意有个空格

print('欢迎您:', user) #输出多项,以,号分隔变量,默认输出结果以空格分隔

print('欢迎您: %s' % user) #输出时变量user里的值代替 %s,注意空格

运行结果: 请输入用户名:hji

欢迎您: hji #注意输出里有个空格

**五 Python变量**

**5.1 变量定义**

可以变化的量,如a = 10,以后还可以改变它, 如 a = 100

与变量相反的是字面量,如字符串 hello, 数字 100

写程序时,如果总是用字面量,就是硬编码.

• 变量名称约定

**第一个字符只能是大小写字母或下划线**

**后续字符只能是大小写字母或数字或下划线**

**区分大小写**

• python是动态类型语言,即不需要预先声明变量的类型

**• 推荐采用的全名方法**

**变量名全部采用小写字母**

**简短、有意义**

**多个单词间用下划线分隔**

**变量名用名词,函数名用谓词(动词+名词)**

**类名采用驼峰形式(每个单词首字母大写)**

**5.2 变量赋值**

• 变量的类型和值在赋值那一刻被初始化

• 变量赋值通过等号来执行,**在使用前必须赋值,不支持 ++ --**

>>> counter = 0

>>> name = 'bob'

• python也支持增量赋值

>>> n += 1 #等价于n = n + 1

>>> n \*= 1 #等价于n = n \* 1

>>> i++

File "<stdin>", line 1

i++

^

SyntaxError: invalid syntax

++a, --a,表示正正为正,负负为正

>>>import this #输出Python的zen

**5.3 运算符**

**• 标准算术运算符**

+ - \* / //(取整) %(取余,模运算) \*\*(幂)

>>> divmod(5,3)

(1, 2)

>>> a, b = divmod(5, 3)

>>> a

1

>>> b

2

**• 比较运算符,得到的结果是True或False**

< <= > >= == != #一个=号是赋值,==号才是比较

**Python支持连续比较**

>>> 10 > 5 > 1

True

>>> 20 > 10 < 15 #相当于20 > 10 and 10 < 15

True

**• 逻辑运算符,得到的结果是True或False**

and not or # not运算优先级最高,and 和 or 运算优先级相同

数据类型: 数字类型: 包含 int,float,bool,complex 四类

序列类型: 包含 string,list,tuple 两类

字典类型: 即 dictionary 一类

**六 数据类型-数字**

**6.1 基本数字类型**

• int: 有符号整数,无小数点

• bool: 布尔值

True:1

False:0

• float: 浮点数,有小数点

• complex: 复数

**tpye(变量名)** #查看变量内数据的类型

**6.2 数字表示方式**

**• python默认以十进制数显示**

**• 数字以0o或0O开头表示为8进制数**

>>> 0o11 9

>>> oct(9) '0o11'

**• 数字以0x或0X开头表示16进制数**

>>> 0x11 17

>>> hex(17) '0x11'

**• 数字以0b或0B开头表示2进制数**

>>> 0b11 3

>>> bin(3) '0b11'

**七 数据类型-序列对象**

**7.1 定义字符串**

• python中字符串被定义为引号之间的字符集合

• python支持使用成对的单引号或双引号

• 无论单引号,还是双引号,表示的意义相同

• python还支持三引号(三个连续的单引号或者双引号),可以用来包含特殊字符

>>> words = '''abc #三引号可以保存用户的输入格式

... def

... cde'''

>>> print(words)

abc

def

cde

>>> words = 'abc\ndef\nghij'

>>> print(words)

abc

def

ghij

• python不区分字符和字符串

>>> py\_str = 'abcdefghaiofwqehriosih'

>>> len(py\_str)

22

**7.2 字符串切片**

• 使用索引运算符[ ]和切片运算符[ 起始下标包含:结束下标不含 ]可得到子字符串

• 第一个字符的索引是0,最后一个字符的索引是-1

• 子字符串包含切片中的起始下标,但不包含结束下标

>>> py\_str ='python'

>>> py\_str[0] #取下标为0的字符

'P'

>>> py\_str[-2] #取下标为-2的字符

'o'

>>> py\_str[2:4] #取下标为2,3的字符串

'th'

>>> py\_str[2:] #取下标为2到结尾的字符串

'thon'

>>> py\_str[:4] #取下标为0到3的字符串

'Pyth'

>>> py\_str[:100] #Python切片,不会提示下标越界的错误

'Python'

>>> py\_str[:] #从开头取到结尾

'Python'

>>> py\_str[::2] #从下标为0的字符开始,以步长2取字符串

'pto' #**字符串名[起始下标(含):结束下标(不含):步长]**

>>> py\_str[1::2] #从下标为1的字符开始,以步长2取字符串

'yhn'

>>> py\_str[::-1] #从下标为-1的字符开始,以步长1取字符串

'nohtyp'

**7.3 字符串连接操作**

• 使用+号可以将多个字符串拼接在一起

• 使用\*号(乘号)可以将一个字符串重复多次

>>> py\_str + ' is good'

'python is good'

>>> py\_str \* 2

'pythonpython'

**7.4 成员关系判断**

>>> 't' in py\_str

True

>>> 'th' in py\_str

True

>>> 'to' in py\_str

False

>>> 'to' not in py\_str

True

**7.5 定义列表**

• 列表以[]定义,它能保存任意数量任意类型的python对象

• 像字符串一样,列表也支持下标和切片操作

• 列表中的项目可以改变

>>> alist = [1,2,3,'tom','jerry']

**7.6 列表操作**

>>> len(alist) #取列表长度

5

>>> alist[0] #取列表下标为0的元素

1

>>> alist[3:] #取列表下标为3到结尾的元素

['tom', 'jerry']

>>> 3 in alist #列表成员关系判断

True

>>> 'o' in alist

False

>>> alist + [4,5] #列表拼接

[1, 2, 3, 'tom', 'jerry', 4, 5]

>>> alist \* 2

[1, 2, 3, 'tom', 'jerry', 1, 2, 3, 'tom', 'jerry']

>>> alist. #按2下TAB键

alist.append( alist.count( alist.insert( alist.reverse(

alist.clear( alist.extend( alist.pop( alist.sort(

alist.copy( alist.index( alist.remove(

>>> alist.append('bob') #向alist某尾追加元素bob

>>> alist

[1, 2, 3, 'tom', 'jerry', 'bob']

**7.6 元组的定义及操作**

• 可以认为元组是“静态”的列表,以()定义

• **元组一旦定义,不能改变,其他操作如切片 成员关系判断等同列表**

>>> atuple = (1,2,3,'tom','jerry')

**八 数据类型-字典**

**8.1 字典的定义及操作**

**• 字典是由键:值(key:value)对构成的映射数据类型,以{}定义,无下标顺序**

**• 通过键取值,不支持下标操作**

>>> adict = {'name': 'tom', 'age': '23'}

>>> len(adict)

2

>>> 'tom' in adict #tom是adict的key吗

False

>>> 'name' in adict #name是adict的key吗

True

>>> adict['name'] #查询键name的键值,查询时用[]号

'tom'

**8.2 数据类型比较**

**•** 按**存储模型**分类

**标量类型:** 数值、字符串 **#数据中不能包含其他类型数据**

**容器类型:** 列表、元组、字典 **#数据中能包含其他类型数据**

**•** 按**更新模型**分类**:**

**可变类型:** 列表、字典

**不可变类型:** 数值、字符串、元组

**•** 按**访问模型**分类 **直接访问:** 数值 **顺序访问:** 字符串、列表、元组 **映射访问:** 字典