# 2.python基本语法

## 2.1.数据类型

数据类型分为基本数据类型和组合数据类型

### 2.1.1.基本数据类型

数值、字符串、布尔类型

**数值**

int 整型   整数

例如：2 18 66 ....

float 浮点型   带小数的数

例如：2.0 18.8 66.666 ....

complex 复数   a+bj

例如：3+4j

**字符串**

str 字符串   视作文本

组成： 由数字、字母、空格、其他字符等组合而成

表达： 用" " 或' '

例如："python 123 @#$^&(())"

**布尔类型**

bool 布尔类型 True、False

主要用于逻辑运算

例如：y = 2 < 1 # y=False

上述类型均可定义单个数据，如果我们有一组数据，该如何表示？

### 2.1.2.组合数据类型

列表、元组、字典、集合

**列表**

list 列表    序列类型:  数据有位置顺序

表示方式：  [data1,data2....]

例如：

a = [1, 2, 3, 4, 5]

a[0] # 1

**元组**

tuple 元组    序列类型

表示方式： (data1,data2...)

元素不支持修改——“不可变的列表”

例如：

b = (1, 2, 3, 4, 5)

b[0] # 1

**字典**

dict 字典  映射类型： 通过“键”-“值”的映射实现数据存储和查找

表示方式： {key1:value1 , key2:value2 , ...}

例如：

student = {201901: "小明", 201902: "小红", 201903: "小强"}

student[201901] # '小明'

**集合**

set 集合  一系列互不相等元素的集合，无序的

表示方式： {data1,data2...}

例如：

s = {"小明", "小红", "小强", "小明"}

s # {'小强', '小明', '小红'}

## 2.2.变量

在程序中，我们如何来引用这些数据？

非常通俗的处理办法：赋值给一个变量

### 2.2.1.变量的概念

“量”   实实在在的对象：如数据、抽象

“变”   可变性：增、删、查、改等

变量定义二要素：  变量名、赋值

例如： x = 1

### 2.2.2.变量的命名

哪些可以用来做变量名？

大写字母、小写字母、数字、下划线、汉字及其组合。

严格区分大小写

例如：

Python\_is\_第1名 = True

python\_is\_第1名 = False

哪些情况不被允许？

首字符不允许为数字

变量名中间不能有空格

不能与33个Pyhton保留字相同

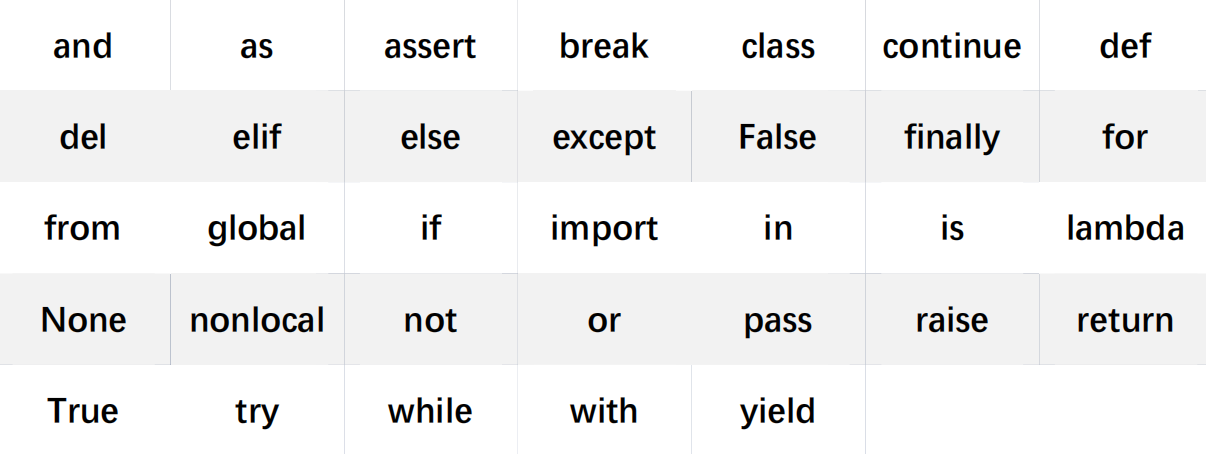
例如：

1\_fruit = "apple"

my fruit = "apple"

if = True

附python保留字



### 2.2.3.变量名定义技巧

变量名尽可能有实际意义，表征数据的某种特性

例如：

a = [17, 18, 19]

age\_of\_students = [17, 18, 19]

下划线（推荐：变量和函数名） 变量名由多个单词组成：用\_连接多个单词

例如：

age\_of\_students = [17, 18, 19]

驼峰体（推荐：类名） 变量名由多个单词组成：单词首字母大写

例如：

AgeOfStudents

尽量避免用中文和拼音做变量名

特殊的变量：常量（不变的量，如π、e）   变量名所有字母均为大写

例如：

MAX\_ITERATION = 1000

### 2.2.4.变量的赋值

一般赋值

通过等号自右向左进行赋值

例如：

x = 1+2

x # 3

增量赋值

例如：

x = 10

x = x+10

x #20

x += 10

打包赋值

例如：

x, y = 1, 2

print(x, y) # 1 2

x, y = y, x

print(x, y) # 2 1

## 2.3.控制流程

### 2.3.1.顺序流程

自上向下依次执行

例如：实现1到5的整数求和

# res = 1+2+3+4+5

res = 0 # 赋初值

res += 1

res += 2

res += 3

res += 4

res += 5

res # 显示结果 15

### 2.3.2.循环流程

**遍历循环（for)**

主要形式：

for 元素 in 可迭代对象：  
  执行语句

执行过程：

从可迭代对象中，依次取出每一个元素，并进行相应的操作

例如：实现1到5的整数求和

res = 0

**for** i **in** [1,2,3,4,5]: *# 每次迭代，取出一个i*

res += i *# 对每次迭代取出的i 进行相应操作*

res *# 遍历结束后，执行后续语句 15*

**无限循环（while）**

主要形式：

while 判断条件：

  条件为真，执行语句

条件为假，while 循环结束

例如：实现1到5的整数求和

while也可以添加else，当while执行完毕后执行else的内容

while 判断条件：

  条件为真，执行语句

else:

while执行完毕，然后执行

i = 1

res = 0

**while** i <= 5: *# 若i不大于5，则循环继续*

res += i

i += 1

res *# 若循环条件不成立，循环停止，执行后续语句 15*

### 2.3.3.分支流程

最简单的形式：

if 判断条件：

  条件为真，执行语句

else:

  条件为假，执行语句

或者也可以：

if 判断条件：

  当前条件为真，执行语句

elif 判断条件：

  当前条件为真，执行语句

else:

  前面条件都为假，执行语句

例如：

age = 18

**if** age > 22:

print("可以结婚啦")

**else**:

print("em，着急了点，再等等。。。") # em，着急了点，再等等。。。

## 2.4.输入和输出

程序中，大量的数据都是从外部输入，同时，最终得到的结果也需要进行输出。

### 2.4.1.数据从哪里来？

1. 外部文件导入

从本地硬盘、网络端读入等

该部分内容放在《文件、异常和模块》进行讲解

2. 程序中定义

例如：

age = 18

name = "Tom"

3. 动态交互输入 input

在程序运行的过程中进行输入

例如：

x = input("请输入一个数字：") # 33

y = input("请输入一个数字：") # 66

print(x + y) # 3366

type(x) #str

eval() 将字符串作为代码执行

例如：

x = eval(input("请输入一个数字：")) # 33

y = eval(input("请输入一个数字：")) # 66

print(x + y) # 99

或者

a = "print('aaa')"

eval(a)

### 2.4.2.数据到哪里去

1. 存储到本地硬盘或网络端

该部分内容放在《文件、异常和模块》进行讲解

2. 打印输出 print

直接打印数据

例如：

print("我是一颗小星星")

打印变量

例如：

x = 1024

print(x)

print 默认换行

例如：

print(1)

print(2)

如果不想换行怎么办？

换行控制 end=

例如：

print(123,end=" ")

print(456)

有时候，我们需要一些复杂的输出:比如几个变量一起组合输出

例如：

PI = 3.1415926

E = 2.71828

print("PI = ", PI, "E = ", E)

3. 格式化输出方法 format

基本格式："字符{ 0 }字符{ 1 }字符".format(v0,v1)

print("PI = **{0}**,E = **{1}**".format(PI, E))

PI = 3.1415926,E = 2.71828

print("PI = **{1}**,E = **{0}**".format(PI, E))

PI = 2.71828,E = 3.1415926

print("PI = **{}**,E = **{}**".format(PI, E))

PI = 3.1415926,E = 2.71828

print("PI = **{0}**,E = **{0}**".format(PI, E))

PI = 3.1415926,E = 3.1415926

再进一步 修饰性输出

填充输出

*# \_\_\_\_3.1415926\_\_\_\_\_ 进行填充*

*#0表示第几个变量，这里就表示第一个变量PI，*

*#冒号“:”表示接下来要对此变量进行修饰，*

*#下划线“\_”表示要填充的图案，*

*#上尖括号“^”表示数据要放的位置，“^”表示数据放中间，“<”表示数据放左边，“>”表示数据放右边*

*#最后的20表示输出的总宽度*

print("**{0:\_^20}**".format(PI))

\_\_\_\_\_3.1415926\_\_\_\_\_\_

print("**{0:\*<30}**".format(PI))

3.1415926\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

数字千分位分隔符

*#显示1,000,000*

print("**{0:,}**".format(10000000))

10,000,000

浮点数简化输出

*#留2位小数*

print("**{0:.2f}**".format(PI))

3.14

*#按百分数输出*

print("**{0:.1%}**".format(0.818727))

81.9%

*#科学计数法输出*

print("**{0:.2e}**".format(0.818727))

8.19e-01

数的进制转换输出

*#十进制整数转二进制、unicode码、十进制、八进制、十六进制输出*

"二进制**{0:b}**,Unicode码**{0:c}**,十进制**{0:d}**,八进制**{0:o}**,十六进制**{0:x}**".format(450)

'二进制111000010,Unicode码ǂ,十进制450,八进制702,十六进制1c2'

## 2.5.注释与缩进

### 2.5.1.注释

单行注释

格式：# 注释内容

a=1 *# 我是单行注释*

多行注释

格式："""注释内容，可分行"""

*"""妈妈*

*我好像*

*发现了*

*写诗的诀窍"""*

### 2.5.2.缩进

用缩进来表示语句间的逻辑

在 for while if def class等 :之后下一行开始进行缩进，表明后续代码与前句之间的从属关系

缩进量：4字符

**for** i **in** [1, 2, 3]:

print(i)

print("打印结束")

### 2.5.3.使用空格

二元运算符两边加一个空格

x = 2 *# 赋值*

x + = 4 *# 增量*

6 > 2 *# 比较*

使用不同优先级的运算符，考虑在最低优先级的运算符周围添加空格

x = x\*2 - 1

z = x\*x + y\*y

c = (a+b) \* (a-b)

在逗号后使用空格

x, y = 1, 2

ls = [1, 2, 3]

不要使用一个以上的空格

x = 2 *# 空格一个就够了，不应过多*

避免使用空格

在制定关键字参数或者默认参数值的时候，不要在=附近加上空格

**def** fun(n=1, m=2):

print(n, m)

最后总结，格式约定的目的：

使大量Python代码风格一致

提升代码可读性

养成良好的编码规范利人利己

上述代码格式参考自PEP8的格式建议

## 2.6.作业练习

题目1（数据类型）

\*列表属于\_\_\_\_\_类型，元组属于\_\_\_\_\_类型，字典属于\_\_\_\_\_类型。

A序列 B映射

\*集合内的元素是否可以重复？

A可以 B不可以

题目2（变量）

\*判断以下变量名命名是否合法，如不合法，请说明原因：

A:boy B：4student C:class D first number

\*请将1，2，3，4，5打包赋值给变量a，b，c，d，e。

\*请分别用下划线命名方式和驼峰体命名方式命名一个变量。

题目3（控制语句）

\*分别简述for循环、while循环和if分支语句的执行过程。

题目4（输入输出）

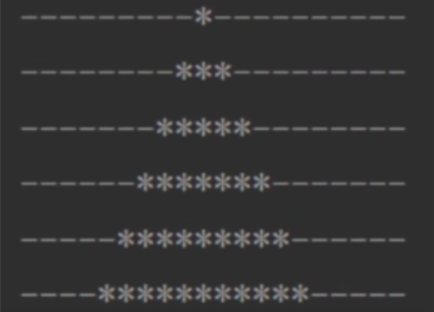
\*用input语句分别输入两个数字，并实现两个数字的相乘。

\*用format分别实现黄金分割比值0.6180339887保留前3位的浮点数输出以及保留1位小数的百分比输出。

\*将下列X处代码补全，实现如图所示的格式化输出。

for i in [0,1,2,3,4,5]:

print("{X}". format("X"\*(2\*i+1))



题目5（程序格式）

\*以下格式是否符合PEP8的格式建议，如不符合，请予以修改

A x\*= 10

B y = (3-1)/8

C t = (1,2,3,4,5)

D x = 10

E def fun(x = 1, y = 2):

F if m > 3:

print(“m大于3”)

答案：

1. \*AAB \*B

2. \*A

\* a,b,c,d,e=1,2,3,4,5

print(a,b,c,d,e)

\* this\_is\_a\_var=2

print(this\_is\_a\_var)

thisIsAVar=3

ThisIsAVar=4

print(thisIsAVar,ThisIsAVar)

1. 略

4. \* num1 =eval（input（“请输入第一个数字：”））

num2 =eval（input（“请输入第二个数字：”））

print（“两个数字相乘的结果是：{}”.format（num1\*num2））

\* # golden\_section=0.6180339887

GOLDEN =0.6180339887

print("{0:.3f}". format(GOLDEN))

print("{0:.1%}". format(GOLDEN))

\* for i in [0,1,2,3,4,5]:

print("{0:-^20}". format("\*" \* (2\*i + 1)))

5. \* A x与\*之间添加一个空格

B 符号之间添加空格：y = ( 3 - 1 ) / 8

C 逗号后面加空格：t = (1, 2, 3, 4, 5)

D 空格太多：x = 10

E 参数的等号两边不需要空格：def fun(x=1, y=2):

F 第二行需要缩进