变量

变量是内存中一个带标签的盒子

In [16]:

- 变量由三部分组成
- 标识:表示对象所存储的内存地址,使用内置函数id(obj)来获取
- 类型:表示的是对象的数据类型,使用内置函数type(obj)来获取
- 值:表示对象所存储的具体数据， 使用print(obj)可以将值进行打印输出

In [18]:

```
name = '玛利亚'
print(name)
print('标识',id(name)) #类型地址
print('类型',type(name))
print('值',name)
```

```
玛利亚
标识 2400914641136
类型 <class 'str'>
值 玛利亚
```

·当多次赋值之后,变量名会指向新的空间

In [23]:

```
name = '玛利亚'#这个将会变成内存垃圾
name = '大佬'
print(name)
```

```
大佬
```

In [1]:

```
name = '玛利亚'
print(name)
name = '大佬'
print(name)
```

```
玛利亚
大佬
```

数据类型

In [24]:

- 常用的数据类型
- 整数类型→int >98
- 浮点数类型→float>3.14159
- 布尔类型>bool →>True ,False # Ture = 1, False = 非 1
- 字符串类型→str →'人生苦短，我用Python'#单双三引号都是字符串类

·整数类型

·可以表示 正数，负数，0 ·表示都是（0（数字）） ((x)（字母）） 十进制→默认的进制 ·二进制→以0b开头 ·八进制→以0o开头 ·十六进制→Ox开头 指针就是16进制

In [3]:

```
n1 = 90
n2 = -12
n3 = 0
print(n1,type(n1))
print(n2,type(n3))
print(n3,type(n3))
```

```
90 <class 'int'>
-12 <class 'int'>
0 <class 'int'>
```

In [28]:

```
print('十进制',118)
print('二进制',0b10101111)#二进制以0b开头
print('八进制',0o176)#八进制以0o开头
print('十六进制',0x1EAF)
```

```
十进制 118
二进制 175
八进制 126
十六进制 7855
```

In [29]:

```
print('八进制',0o186)            #八进制不能超过八
```

```
  File "<ipython-input-29-ad6d30171b62>", line 2
    print('八进制',0o186)
                    ^
SyntaxError: invalid digit '8' in octal literal
```

In [5]:

```
print('二进制',0b10101111)        #是0（数字）不是O(字母)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call last)
Input In [5], in <cell line: 1>()
----> 1 print('二进制',0b10101111)

NameError: name '0b10101111' is not defined
```

浮点类型

In [ ]:

- 浮点数整数部分和小数部分组成
- 浮点数存储不精确性
- 使用浮点数进行计算时，可能会出现小数位数不确定的情况

In [1]:

```
-3.14e2 # 即-314
3.14e-3 # 即0.00314
```

Out[1]:

```
0.00314
```

In [2]:

```
print(3.14e500)
print(-3.14e500)
```

```
inf
-inf
```

In [11]:

```
#整数加浮点数等于浮点数

#运算变成更宽的类型

x = 1.1 + 2
print(x)
print(type(x))
a = 3.14159
print(a,type(a))
```

```
3.1
<class 'float'>
3.14159 <class 'float'>
```

浮点数不精确

In [41]:

```
n1 = 1.1
n2 = 2.2
n3 = 2.1
print(n1+n2)
print(n1+n3)#不是所有的不精确
```

```
3.3000000000000003
3.2
```

解决方案·导入模块decimal

In [49]:

```
from decimal import Decimal
print(Decimal('1.1') +Decimal('2.2'))
```

```
3.3
```

布尔类型

In [ ]:

```
•用来表示真或假的值
• True表示真，False表示假
•布尔值可以转化为整数
• True→1
• False→0
•以下对象的布尔值为False
False
数值0
None
空字符串
空列表
空元组
空字典
空集合
```

In [6]:

```
f1 = True
f2 = False
print(f1,type(f1))
print(f2,type(f2))
print("------------以下对象的布尔值均为True------------")
print(bool(False)) #False
print(bool(0))     #False
print(bool(0.0))   #False
print(bool (None)) #False
print(bool(''))   #False 空字符串
print(bool(""))   #False 空字符串
print(bool([]))      #空列表
print(bool(list()))  #空列表
print(bool(()))      #空元祖
print(bool(tuple())) #空元祖
print(bool({}))      #空字典
print(bool(dict()))  #空字典
print(bool(set()))   #空集合
print("------------其它对象的布尔值均为True------------")
```

```
True <class 'bool'>
False <class 'bool'>
False
False
False
False
False
False
False
False
False
False
False
False
False
False
False
------------其它对象的布尔值均为True------------
```

布尔值可以转化整数

In [51]:

```
print(f1)#这样不行
```

```
True
```

In [53]:

```
print(f1+1)#1+1的结果为2，True表示1
print(f2+0)#0+0的结果为0，False表示0
```

```
2
0
```

字符串类型

In [ ]:

- 字符串又被称为不可变的字符序列
- 可以使用单引号’ 双引号"" 三引号""或"" ’ """来定义
- 单引号和双引号定义的字符串必须在一行
- 三引号定义的字符串可以分布在连续的多行

In [3]:

```
str1 = '人生苦短，why not a try'
str2 = "why not a try"
str3 = '''人生苦短，人生苦短，why not a try'''
str4= '''人生苦短，
why not a try'''
str5 = """人生苦短，why not a try"""
str6 = """人生苦短，
why not a try"""

print(str1,type(str1))
print(str2,type(str2))
print(str3,type(str3))
print(str4,type(str4))
print(str5,type(str5))
print(str6,type(str6))
```

```
人生苦短，why not a try <class 'str'>
why not a try <class 'str'>
人生苦短，人生苦短，why not a try <class 'str'>
人生苦短，
why not a try <class 'str'>
人生苦短，why not a try <class 'str'>
人生苦短，
why not a try <class 'str'>
```

In [61]:

```
str1 = ''人生苦短，why not a try''
str1 =""人生苦短，why not a try"" #不能用两个单引号或者双引号
str2 = "人生苦短，
why not a try"
str1 = '人生苦短，
why not a try'#单或者双引号不能换行
```

```
  File "<ipython-input-61-b548ed6666ec>", line 1
    str1 = ''人生苦短，why not a try''
                ^
SyntaxError: invalid character in identifier
```

复数

In [5]:

```
z = 1.23e+89j
print(type(z))
c1 = 12 + 0.2j
c2 = 6 - 1.2j
#对负数进行运算
print("c1+c2",c1+c2)
print("c1*c2",c1*c2)
```

```
<class 'complex'>
c1+c2 (18-1j)
c1*c2 (72.24-13.2j)
```

In [ ]:

我们把形如z=a+bi(ab均为实教)的数称为复数,其中a称为实部，b称为虚部，i为虚数单位。
当z的虚部等于零时,常称z为实数;当z的虚部不等于零时，实部等于零时，常称z为纯虚数
复数域是实s域的代教闭包，即任何复系教多项式在复教域中总有根。
a，b都是浮点类型，虚数部分用j或者J标识
complcx(rcal,imag) 创建复数

In [5]:

```
complex_one = 1 + 2j # 实部为 1，虚部为 2
complex_two = 2j # 实部为 0，虚部为 2
complex_one = 1 + 2j
print(complex_one.real) # 获取复数实部
print(complex_one.imag) # 获取复数虚部
```

```
1.0
2.0
```

ptyhon中的注释

In [ ]:

• 在代码中对代码的功能进行解释说明的标注性文字，可以提高代码的可读性
• 注释的内容会被Python解释器忽略
通常包括三种类型的注释
• 单行注释→以"#"开头,直到换行结束
• 多行注释>并没有单独的多行注释标记，将一对三引号之间的代码称为多行注释
• 中文编码声明注释→在文件开头加上中文声明注释．用以指定源码文件的编码格式

In [111]:

```
 #输入功能（单行注释）
print('hello')
'''嘿嘿
我是
多行注释'''
```

```
hello
```

Out[111]:

```
'嘿嘿\n我是\n多行注释'
```

数据类型转换

引入

In [16]:

```
print(int(4.5))#整形去掉小数部分
print(float(3))#加小数部分
```

```
4
3.0
```

In [2]:

```
a = input('请输入一个加数：')
b = input('请输入一个加数：')
print(type(a),type(b))
print(a + b)

#这里str加str类型
```

```
请输入一个加数：10
请输入一个加数：20
<class 'str'> <class 'str'>
1020
```

In [1]:

```
#从键盘录入两个整数，计算两个整数的和
a = input()
b = input()
print(type(a),type(b))
print(a + b)
```

```
10
20
<class 'str'> <class 'str'>
1020
```

input输入的就是字符串，即使输入0也不是False而是字符串

In [1]:

```
# a = input('请输入一个加数：')
a = input('请输入一个加数：')
a = int(a)#将转换之后的结果存储到a中
b = input('请输入一个加数：')
b = int(b)
print(type(a),type(b))
print(a+b)
```

```
请输入一个加数：10
请输入一个加数：20
<class 'int'> <class 'int'>
30
```

In [6]:

```
name = '狂徒张三'
age = 20

print(type(name),type(age))#说明name 与age的数据类型不相同
```

```
<class 'str'> <class 'int'>
```

In [8]:

```
print('我叫'+ name +'今年'+str(age)+'岁')#将int类型通过str()函数转成了str类型
```

```
我叫狂徒张三今年20岁
```

In [7]:

```
print('我叫'+ name +'今年'+age+'岁') #这里的 + 是连接符

#当将str类型与int类型进行连接时，报错，解决方案，类型转换，报错需要类型转换
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-7-fefc5d8134e2> in <module>
----> 1 print('我叫'+ name +'今年'+age+'岁') #这里的 + 是连接符
      2
      3 #当将str类型与int类型进行连接时，报错，解决方案，类型转换，报错需要类型转换

TypeError: can only concatenate str (not "int") to str
```

字符串类型转化 &&几乎其他类型都能转字符串类型###

In [9]:

```
print('----------str()将其他类型转化成str类型--------')
a = 10
b = 198.8
c = False

print(type(a),type(b),type(c))
print(str(a),str(b),str(c),type(str(a)),type(str(b)),type(str(c)))
```

```
----------str()将其他类型转化成str类型--------
<class 'int'> <class 'float'> <class 'bool'>
10 198.8 False <class 'str'> <class 'str'> <class 'str'>
```

int类型转化

In [10]:

```
print('----------int()将其他类型转化成int类型--------')
s1 = '128'
f1 = 98.7
s2 = '76.77'
f2 = True
s3 = 'hellow'

print(type(s1),type(f1),type(s2),type(f2),type(s3))
print(int(s1),type(int(s1)))#将str转成int类型,字符串为数字串
print(int(f1), type(int(f1)))#float转成int类型,截取整数部分,s,舍掉小数部分
print(int(f2), type(int(f2)))#布尔类型可以转化成int
```

```
----------int()将其他类型转化成int类型--------
<class 'str'> <class 'float'> <class 'str'> <class 'bool'> <class 'str'>
128 <class 'int'>
98 <class 'int'>
1 <class 'int'>
```

In [11]:

```
rint(int(s2),type(int(s2)))
#将str转成int类型，报错,因为字符串为小数串
print(int(s3),type(int(s3)))
#将str转成int类型时，字符串必须为数字串（整数），非数字串是不允许转换
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-eb66ded23e7e> in <module>
----> 1 rint(int(s2),type(int(s2)))
      2 #将str转成int类型，报错,因为字符串为小数串
      3 print(int(s3),type(int(s3)))
      4 #将str转成int类型时，字符串必须为数字串（整数），非数字串是不允许转换

NameError: name 'rint' is not defined
```

float类型转化

In [12]:

```
print('--------float()将其他类型转化成float类型--------')
s1 = '128.87'
f1 = 98
s2 = '76'
f2 = True
s3 = 'hellow'

print(type(s1),type(f1),type(s2),type(f2),type(s3))
print(float(s1),type(float(s1)))
print(float(f1),type(float(f1)))
print(float(s2),type(float(s2)))
print(float(f2),type(float(f2)))
```

```
--------float()将其他类型转化成float类型--------
<class 'str'> <class 'int'> <class 'str'> <class 'bool'> <class 'str'>
128.87 <class 'float'>
98.0 <class 'float'>
76.0 <class 'float'>
1.0 <class 'float'>
```

In [13]:

```
print(float(s3),type(float(s3)))
#字符串中的数据如果是非数字串，则不允许转换
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-2e4229151876> in <module>
----> 1 print(float(s3),type(float(s3)))
      2 #字符串中的数据如果是非数字串，则不允许转换

ValueError: could not convert string to float: 'hellow'
```

ord()、chr()、str()

In [1]:

```
print(ord('a'))              # 返回字符的ASCII码
print(ord('董'))             # 返回汉字字符的Unicode编码
print(chr(65))               # 返回指定ASCII码对应的字符
print(chr(33891))            # 返回指定Unicode编码对应的汉字
print(str([1, 2, 3, 4]))     # 把列表转换为字符串
print(str({1, 2, 3, 4}))     # 把集合转换为字符串
```

```
97
33891
A
董
[1, 2, 3, 4]
{1, 2, 3, 4}
```

bin()、oct()、hex()

In [2]:

```
print(bin(8888))            # 把整数转换为二进制
print(oct(8888))            # 把整数转换为八进制
print(hex(8888))            # 把整数转换为十六进制
```

```
0b10001010111000
0o21270
0x22b8
```

字符串

使用%格式化字符串

%s	字符串 (采用str()的显示)
%r	字符串 (采用repr()的显示)
%c	单个字符
%b	二进制整数
%d	十进制整数
%i	十进制整数
%o	八进制整数
%x	十六进制整数
%e	指数 (基底写为e)
%E	指数 (基底写为E)
%f	浮点数
%F	浮点数，与上相同
%g	指数(e)或浮点数 (根据显示长度)
%G	指数(E)或浮点数 (根据显示长度)
%%	字符"%"

与%d等符号其实没什么关系

In [15]:

```
value = 10
format = '我今年%d岁。'
print(format % value)
```

我今年10岁。

In [3]:

```
value = 10
a = 11
format = '我今年%d岁。'
print(format % a)
```

我今年11岁。

In [9]:

```
value = '周一'
format = '今天是%d'
print(format % value) #周一是字符串类型
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call last)
<ipython-input-9-6e01f263f536> in <module>
      1 value = '周一'
      2 format = '今天是%d'
----> 3 print(format % value) #周一是字符串类型

TypeError: %d format: a number is required, not str
```

In [8]:

```
value = '周一'
format = '今天是%s'
print(format % value)
```

今天是周一

In [3]:

```
name = '张倩'
age = 27
address = '北京市昌平区'
print('----------------------')
print("姓名：%s" % name)
print("年龄：%d岁\n家庭住址：%s" % (age, address))
print('----------------------')
```

```
----------------------
姓名：张倩
年龄：27岁
家庭住址：北京市昌平区
----------------------
```

使用format.....

format常见用法包括格式化填充字符,文本的填充与对齐,格式转换等.

In [10]:

```
name = '张倩'
string = "姓名：{}"
print(string.format(name))
```

姓名：张倩

In [11]:

```
name = '张倩'
age = 25
string = "姓名：{}\n年龄：{}"
print(string.format(name, age))
```

姓名：张倩
年龄：25

In [12]:

```
name = '张倩'
age = 25
string = "姓名：{1}\n年龄：{0}"
print(string.format(age, name))
```

姓名：张倩
年龄：25

In [14]:

```
name = '张倩'
age = 25
weight = 65
string = "姓名：{name}\n年龄：{age}\n体重：{weight}kg"
print(string.format(name=name, weight=weight, age=age))
```

姓名：张倩
年龄：25
体重：65kg

In [15]:

```
points = 19
total = 22
print('所占百分比：{:.2%}'.format(points/total))  # 保留两位小数
```

所占百分比：86.36%

In [6]:

```
print("""
注意你的{},他会变成你的{};
注意你的{},他会变成你的{};
注意你的{},他会变成你的{};
注意你的{},他会变成你的{};
注意你的{},他会变成你的{}
""".format("思想","言语","言语","行动","行动","习惯","习惯","性格","性格","命运"))
```

注意你的思想,他会变成你的言语;
注意你的言语,他会变成你的行动;
注意你的行动,他会变成你的习惯;
注意你的习惯,他会变成你的性格;
注意你的性格,他会变成你的命运

In [9]:

```
print('{:.2f}'.format(3.1415926)) #保留两位小数
print('{:.0f}'.format(3.1415926)) #不保留
print('{:+.2f}'.format(-3.1415926)) #不管是不是正号负号，都在前面加上“+”
print('{:.2%}'.format(0.25)) #保留百分号位
print('{:0>10d}'.format(100)) #用0填充补齐的位置（右对齐）
print('{:,}'.format(1234567890)) #冒号后边加逗号,形成千位分隔符
print('{0:_},{0:#_x}'.format(10000000))# _表示在数字中插入下画线作为千分符
# 字符串前面加字符f，Python 3.6之后的版本支持这种用法
width = 8
height = 6
print(f'Rectangle of {width}*{height}\nArea:{width*height}')
# 格式化为百分数字符串，总宽度为10，保留2位小数，>表示右对齐
print('{0:>10.2%}'.format(1/3))
# 逗号表示在数字字符串中插入逗号作为千分符，#x表示格式化为十六进制数
print("{0:,} in hex is: {0:#x}, in oct is {0:#o}".format(5555555))
# 可以先格式化下标为1的参数，再格式化下标为0的参数
# o表示八进制数，
'''但不带前面的引导符'0o'''
print("{1} in hex is: {1:#x}, {0} in oct is {0:o}".format(6666, 66666))
```

```
3.14
3
-3.14
25.00%
0000000100
1,234,567,890
10_000_000,0x98_9680
Rectangle of 8*6
Area:48
    33.33%
5,555,555 in hex is: 0x54c563, in oct is 0o25142543
66666 in hex is: 0x1046a, 6666 in oct is 15012
```

encode()

In [10]:

```
bookName = '《Python可以这样学》'
print(bookName.encode())
print(bookName.encode('gbk'))
print(bookName.encode('gbk').decode('gbk'))
```

```
b'\xe3\x80\x8aPython\xe5\x8f\xaf\xe4\xbb\xa5\xe8\xbf\x99\xe6\xa0\xb7\xe5\xad\xa6\xe3
\x80\x8b'
b'\xa1\xb6Python\xbf\xc9\xd2\xd4\xd5\xe2\xd1\xf9\xd1\xa7\xa1\xb7'
《Python可以这样学》
```

index()、rindex()、count()

In [11]:

```
text = '处处飞花飞处处；声声笑语笑声声。'
print(text.rindex('处'))


print(text.index('声'))
print(text.count('处'))
```

```
6
8
4
```

maketrans()、translate()

In [12]:

```
table = ''.maketrans('0123456789', '零一二三四伍陆柒捌玖')
print('Tel:30647359'.translate(table))
```

```
Tel:三零陆四柒三伍玖
```

ljust()、rjust()、center()

In [13]:

```
print('居左'.ljust(20)+'结束')
print('居右'.rjust(20, '#'))        # 左侧使用井号填充
print('居中'.center(20, '='))       # 两侧使用等号填充
```

```
居左                  结束
##################居右
=========居中=========
```

f-string格式化字符串

In [16]:

```
age = 20
gender = '男'
print(f'年龄：{age}，性别：{gender}')
```

年龄：20，性别：男

split()、rsplit()、join()

In [14]:

```
text = 'Beautiful is better than ugly.'
print(text.split())                     # 使用空白字符进行分隔
print(text.split(maxsplit=1))           # 最多分隔一次
print(text.rsplit(maxsplit=2))          # 最多分隔两次
print('1,2,3,4'.split(','))            # 使用逗号作为分隔符
print(','.join(['1', '2', '3', '4']))   # 使用逗号作为连接符
print(':'.join(map(str, range(1, 5))))  # 使用冒号作为连接符
print(''.join(map(str, range(1, 5))))   # 直接连接，不插入任何连接符
```

```
['Beautiful', 'is', 'better', 'than', 'ugly.']
['Beautiful', 'is better than ugly.']
['Beautiful is better', 'than', 'ugly.']
['1', '2', '3', '4']
1,2,3,4
1:2:3:4
1234
```

lower()、upper()、capitalize()、title()、swapcase()

In [15]:

```
text = 'Explicit is better than implicit.'
print(text.lower())
print(text.upper())
print(text.capitalize())
print(text.title())
print(text.swapcase())
```

```
explicit is better than implicit.
EXPLICIT IS BETTER THAN IMPLICIT.
Explicit is better than implicit.
Explicit Is Better Than Implicit.
eXPLICIT IS BETTER THAN IMPLICIT.
```

startswith()、endswith()

In [20]:

```
text = 'Simple is better than complex.'
print(text.startswith('simple'))
print(text.startswith('Simple'))
print(text.endswith(('.', '!', '?')))
text = 'Simple is better than complex!'
print(text.endswith(('.', '!', '?')))
```

```
False
True
True
False
```

strip()、rstrip()、lstrip()

In [21]:

```
text = '    ======test===#####   '
print(text.strip())          # 删除两侧的空白字符
print(text.strip('=# '))     # 删除两侧的=、#和空格
```

```
======test===#####
test
```

多个类型转化（拓展）

举例项目中（数据分析）的量钢化处

In [16]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['font.sans-serif'] = ['SimHei'] # 中文显示
plt.rcParams['axes.unicode_minus'] = False
```

In [18]:

```
columns = ['age','sex','bmi','children','smoker','region','charges']
df = pd.read_csv("insurance.csv",names = columns)
df
```

Out[18]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

In [20]:

```
#处理sex分类：女性为0，男性为1
df1 = df.replace(['female','male'],[0,1])
#处理region分类：northwest = 0, northeast = 1, southwest = 2, southeast = 3
df2 = df1.replace(['northwest','northeast','southwest','southeast'],[1, 2, 3, 4])
#处理smoker分类：yes = 1,no = 0
df3 = df2.replace(['no','yes'],[0,1])
df3
#保险项目举例
```

Out[20]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	4	1725.55230
2	28	1	33.000	3	0	4	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	0	1	10600.54830
1334	18	0	31.920	0	0	2	2205.98080
1335	18	0	36.850	0	0	4	1629.83350
1336	21	0	25.800	0	0	3	2007.94500
1337	61	0	29.070	0	1	1	29141.36030

1338 rows × 7 columns

可变类型：在id(内存地址)不变的情况下，value（值）可以变，则称为可变类型

不可变类型：value(值)一旦改变，id（内存地址）也改变，则称为不可变类型（id变，意味着创建了新的内存空间)

一、python中数据类型（红色为可变类型）

1、字符串 str
2、布尔类型 bool
3、整数 int
4、浮点数 float
5、数字 (int和float)
6、列表 list
7、元组 tuple
8、字典 dict
9、日期 date

In [ ]: