第一部分 数据类型

1. 基本类型：数字、字符串、布尔

1.1 数字类型

- int 整型　整数

In [1]:

```
2
```

Out[1]:

2

- float 浮点型　带小数的数

In [ ]:

```
2.0
```

- complex 复数　a+bj

In [ ]:

```
3+4j
```

1.2 字符串类型

- 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' '

In [44]:

```
"python 123 @#$^&(())"
```

Out[44]:

'python 123 @#$^&(())'

1.3 布尔类型

- bool 布尔类型
- 主要用于逻辑运算

In [47]:

```
y = 2 < 1
y
```

Out[47]:

```
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. 组合类型：列表、元组、字典、集合

2.1 列表

- list **列表**　　**序列类型：数据有位置顺序**
- **表示方式：** [data1,data2....]

In [48]:

```
a = [1, 2, 3, 4, 5]
a[0]
```

Out[48]:

```
1
```

2.2 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”

In [49]:

```
b = (1, 2, 3, 4, 5)
b[0]
```

Out[49]:

```
1
```

2.3 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 , ...}

In [54]:

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

Out[54]:

'小明'

2.4 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...}

In [55]:

```
s = {"小明", "小红", "小强", "小明"}
s
```

Out[55]:

{'小强', '小明', '小红'}

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

第二部分 变量

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**

In [ ]:

```
x = 1
```

2. 变量的命名

2.1 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

In [58]:

```
Python_is_第1名 = True
python_is_第1名 = False
```

2.2 哪些情况不被允许?

- 首字符不允许为数字

In [59]:

```
1_fruit = "apple"
```

```
  File "<ipython-input-59-e3b1d93d01a0>", line 1
    1_fruit = "apple"
     ^
SyntaxError: invalid token
```

- 变量名中间不能有空格

In [60]:

```
my fruit = "apple"
```

```
  File "<ipython-input-60-36327c3a601f>", line 1
    my fruit = "apple"
          ^
SyntaxError: invalid syntax
```

- 不能与33个Pyhton保留字相同

In [62]:

```
if = True
```

```
  File "<ipython-input-62-4c75bcfb9bb9>", line 1
    if = True
       ^
SyntaxError: invalid syntax
```

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

In [ ]:

```
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**

In [ ]:

```

```

- 驼峰体（推荐：类名） **变量名由多个单词组成：单词首字母大写**

In [ ]:

```
AgeOfStudents
```

- 尽量避免用中文和拼音做变量名

In [ ]:

```

```

- 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**

In [ ]:

```
MAX_ITERATION = 1000
```

3. 变量的赋值

3.1 一般赋值

- 通过等号自右向左进行赋值

In [77]:

```
x = 1+2
x
```

Out[77]:

3

3.2 增量赋值

In [3]:

```
x = 10
x = x+10
x
x += 10
```

Out[3]:

20

3.3 打包赋值

In [5]:

```
x, y = 1, 2
print(x, y)
x, y = y, x
print(x, y)
```

1 2
2 1

第三部分 控制流程

1. 顺序流程

- 自上向下依次执行

【小例子】实现1到5的整数求和

In [6]:

```
# 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
```

Out[6]:

15

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

【小例子】实现1到5的整数求和

In [7]:

```
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
```

Out[7]:

15

3. 循环流程——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**

【小例子】实现1到5的整数求和

In [8]:

```
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
```

Out[8]:

15

4. 分支流程（if）

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

In [9]:

```
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。。。
```

- 有了数据和变量，以及控制流程这些个中间过程后
- 我们回过头来考虑下**程序的输入和输出**

第四部分 输入输出

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 第八章《文件、异常和模块》进行讲解

2. 程序中定义

In [ ]:

```
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

In [10]:

```
x = input("请输入一个数字：")
x
```

```
请输入一个数字：4
```

Out[10]:

```
'4'
```

In [12]:

```
y = input("请输入一个数字：")
y
```

请输入一个数字：3.5

Out[12]:

'3.5'

In [13]:

```
x + y
```

Out[13]:

'43.5'

In [14]:

```
type(x)
```

Out[14]:

str

- eval() 去掉引号

In [15]:

```
x = eval(input("请输入一个数字："))
x
```

请输入一个数字：4

Out[15]:

4

In [16]:

```
y = eval(input("请输入一个数字："))
y
```

请输入一个数字：3.5

Out[16]:

3.5

In [17]:

```
x + y
```

Out[17]:

7.5

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 第八章《文件、异常和模块》进行讲解

2. 打印输出 print

- 直接打印数据

In [18]:

```
print("我是一颗小星星")
```

我是一颗小星星

In [19]:

```
print(1234)
```

1234

- 打印变量

In [20]:

```
x = 1024
print(x)
```

1024

- print 默认换行

In [21]:

```
print(1)
print(2)
```

1
2

- 如果不想换行怎么办?

- 换行控制 end=

In [24]:

```
print(123,end=" ")
print(456)
```

```
123 456
```

- 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

In [25]:

```
PI = 3.1415926
E = 2.71828
print("PI = ", PI, "E = ", E)
```

```
PI =  3.1415926 E =  2.71828
```

3. 格式化输出方法 format

- **基本格式："字符{ 0 }字符{ 1 }字符".format(v0,v1)**

In [26]:

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

```
PI = 3.1415926,E = 2.71828
```

In [27]:

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

```
PI = 2.71828,E = 3.1415926
```

In [28]:

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

```
PI = 3.1415926,E = 2.71828
```

In [29]:

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

```
PI = 3.1415926,E = 3.1415926
```

- 再进一步 修饰性输出

1. 填充输出

In [31]:

```
# ____3.1415926_____  进行填充
print("{0:_^20}".format(PI))
```

_____3.1415926_____

In [32]:

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

3.1415926**********

2. 数字千分位分隔符

- 显示1,000,000

In [33]:

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

10,000,000

In [34]:

```
print("{0:&>20,}".format(10000000))
```

&&&&&&&&&&10,000,000

In [35]:

```
print("{0:,&>20}".format(10000000))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-35-9f30412a92d9> in <module>
----> 1 print("{0:,&>20}".format(10000000))

ValueError: Invalid format specifier
```

3. 浮点数简化输出

- 留2位小数

In [36]:

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

3.14

- 按百分数输出

In [39]:

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

81.9%

- 科学计数法输出

In [40]:

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

8.19e-01

4. 整数的进制转换输出

- 十进制整数转二进制、unicode码、十进制、八进制、十六进制输出

In [42]:

```
"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)
```

Out[42]:

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

第五部分 程序格式

1. 行最大长度

所有行限制的最大字符数为79

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

In [36]:

```
for i in [1, 2, 3]:
    print(i)
print("打印结束")
```

3. 使用空格

- 二元运算符两边加一个空格

In [ ]:

```
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

In [ ]:

```
x = x*2 - 1
z = x*x + y*y
c = (a+b) * (a-b)
```

- 在逗号后使用空格

In [44]:

```
x, y = 1, 2
ls = [1, 2, 3]
```

- 不要使用一个以上的空格

In [ ]:

```
x      = 2          # 空格一个就够了，不应过多
```

4. 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

In [45]:

```
def fun(n=1, m=2):
    print(n, m)
```

小结

1、以上属于PEP8格式指南的部分内容，养成良好的编码规范利人利己

2、格式约定的目的：

- 使大量Python代码风格一致
- 提升代码可读性

3、尽信书不如无书，不应死板教条的执行格式规范

- 项目规范优先

5. 注释

- 单行注释
- 格式：# 注释内容

In [37]:

```
a=1  # 我是单行注释
```

- 多行注释
- 格式："""注释内容，可分行"""

In [ ]:

```
"""
妈妈
我
好像
发现了
写诗
的
诀窍
"""
```