第二章 第一节 Python 基本数据类型

- **Python 数据类型**

内置数据类型

在编程中，数据类型是一个重要的概念。

变量可以存储不同类型的数据，并且不同类型可以执行不同的操作。

在这些类别中，Python 默认拥有以下内置数据类型：

文本类型:	str
数值类型:	int, float, complex
序列类型:	list, tuple, range
映射类型:	dict
集合类型:	set, frozenset
布尔类型:	bool
二进制类型:	bytes, bytearray, memoryview

第二章 第一节 Python 基本数据类型

- **Python 数据类型**

获取数据类型

您可以使用 type() 函数获取任何对象的数据类型：

实例

打印变量 x 的数据类型：

```
x = 10
print(type(x))
```

设置数据类型

在 Python 中，当您为变量赋值时，会设置数据类型：

第二章 第一节 Python 基本数据类型

- **Python 数据类型**

设置数据类型

在 Python 中，当您为变量赋值时，会设置数据类型：

示例	数据类型	
x = "Hello World"	str	
x = 29	int	
x = 29.5	float	
x = 1j	complex	
x = ["apple", "banana", "cherry"]	list	
x = ("apple", "banana", "cherry")	tuple	
x = range(6)	range	
x = {"name" : "Bill", "age" : 63}	dict	
x = {"apple", "banana", "cherry"}	set	
x = frozenset({"apple", "banana", "cherry"})	frozenset	
x = True	bool	
x = b"Hello"	bytes	
x = bytearray(5)	bytearray	
x = memoryview(bytes(5))	memoryview	

第二章 第一节 Python 基本数据类型

- **Python 数据类型**

设置特定数据类型

如果希望指定数据类型，则您可以使用以下构造函数：

示例	数据类型	
x = str("Hello World")	str	
x = int(29)	int	
x = float(29.5)	float	
x = complex(1j)	complex	
x = list(("apple", "banana", "cherry"))	list	
x = tuple(("apple", "banana", "cherry"))	tuple	
x = range(6)	range	
x = dict(name="Bill", age=36)	dict	
x = set(("apple", "banana", "cherry"))	set	
x = frozenset(("apple", "banana", "cherry"))	frozenset	
x = bool(5)	bool	
x = bytes(5)	bytes	
x = bytearray(5)	bytearray	
x = memoryview(bytes(5))	memoryview	

第二章 第一节 Python 基本数据类型

- **Python 数字**

  Python 中有三种数字类型：

  int

  float

  complex

  为变量赋值时，将创建数值类型的变量：

  实例

  ```
  x = 10   # int
  y = 6.3  # float
  z =2j   # complex
  ```

  Int

  Int 或整数是完整的数字，正数或负数，没有小数，长度不限。

  NumPy uses fixed-bit data types, such as int32 (32-bit integer) and int64 (64-bit integer).

  float

  Positive, negative and 0 float.

  1.23e-4表示0.000123

第二章 第一节 Python 基本数据类型

- **Python 数字**

  复数

  复数用 "j" 作为虚部编写：

  实例

  复数：

```
x = 2+3j
y = 7j
z = -7j
```

  类型转换

  您可以使用 int()、float() 和 complex() 方法从一种类型转换为另一种类型：

  实例

  从一种类型转换为另一种类型：

```
x = 10 # int
y = 6.3 # float
z = 1j # complex
# 把整数转换为浮点数
a = float(x)
# 把浮点数转换为整数
b = int(y)
# 把整数转换为复数：
c = complex(x)
```

  注释：您无法将复数转换为其他数字类型。

第二章 第一节 Python 基本数据类型

- **Python 数字**

  随机数

  Python 没有 random() 函数来创建随机数，但 Python 有一个名为 random 的内置模块，可用于生成随机数：

  实例

  导入 random 模块，并显示 1 到 9 之间的随机数：

```
import random
print(random.randrange(1,10))
```

第二章 第一节 Python 基本数据类型

- **Python Casting**

  指定变量类型

  有时您可能需要为变量指定类型。这可以通过 casting 来完成。 Python 是一门面向对象的语言，因此它使用类来定义数据类型，包括其原始类型。

  因此，使用构造函数完成在 python 中的转换：

  int() - 用整数字面量、浮点字面量构造整数（通过对数进行下舍入），或者用表示完整数字的字符串字面量

  float() - 用整数字面量、浮点字面量，或字符串字面量构造浮点数（提供表示浮点数或整数的字符串）

  str() - 用各种数据类型构造字符串，包括字符串，整数字面量和浮点字面量

  实例

  整数：

  ```
  x = int(1)   # x 将是 1
  y = int(2.5) # y 将是 2
  z = int("3") # z 将是 3
  ```

第二章 第一节 Python 基本数据类型

- **Python Casting**

  指定变量类型

  有时您可能需要为变量指定类型。这可以通过 casting 来完成。 Python 是一门面向对象的语言，因此它使用类来定义数据类型，包括其原始类型。

  因此，使用构造函数完成在 python 中的转换：

  int() - 用整数字面量、浮点字面量构造整数（通过对数进行下舍入），或者用表示完整数字的字符串字面量

  float() - 用整数字面量、浮点字面量，或字符串字面量构造浮点数（提供表示浮点数或整数的字符串）

  str() - 用各种数据类型构造字符串，包括字符串，整数字面量和浮点字面量

  实例

  整数：

  ```
  x = int(1)   # x 将是 1
  y = int(2.5) # y 将是 2
  z = int("3") # z 将是 3
  ```

  实例

  浮点数：

  ```
  x = float(1)     # x 将是 1.0
  y = float(2.5)   # y 将是 2.5
  z = float("3")   # z 将是 3.0
  w = float("4.6") # w 将是 4.6
  ```

第二章 第一节 Python 基本数据类型

- **Python Casting**

  实例

  字符串：

  ```
  x = str("S2") # x 将是 'S2'
  y = str(3)    # y 将是 '3'
  z = str(4.0)  # z 将是 '4.0’
  ```

- **Python 字符串**

  字符串字面量

  python 中的字符串字面量由单引号或双引号括起。

  'hello' 等同于 "hello"。

  您可以使用 print() 函数显示字符串字面量：

  实例

  ```
  print("Hello")
  print('Hello')
  ```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  多行字符串

  您可以使用三个引号将多行字符串赋值给变量：

  实例

  您可以使用三个双引号：

  ```
  a = """Python is a widely used general-purpose, high level programming language.
  It was initially designed by Guido van Rossum in 1991
  and developed by Python Software Foundation.
  It was mainly developed for emphasis on code readability,
  and its syntax allows programmers to express concepts in fewer lines of code."""
  print(a)
  ```

  或三个单引号：

  实例

  ```
  a = '''Python is a widely used general-purpose, high level programming language.
  It was initially designed by Guido van Rossum in 1991
  and developed by Python Software Foundation.
  It was mainly developed for emphasis on code readability,
  and its syntax allows programmers to express concepts in fewer lines of code.'''
  print(a)
  ```

  注释：在结果中，换行符插入与代码中相同的位置。

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  多行字符串数组

  像许多其他流行的编程语言一样，Python 中的字符串是表示 unicode 字符的字节数组。

  但是，Python 没有字符数据类型，单个字符就是长度为 1 的字符串。

  方括号可用于访问字符串的元素。

  实例

  获取位置 1 处的字符（请记住第一个字符的位置为 0）：

  ```
  a = "Hello, World!"
  print(a[1])
  ```

  裁切

  您可以使用裁切语法返回一定范围的字符。

  指定开始索引和结束索引，以冒号分隔，以返回字符串的一部分。

  实例

  获取从位置 2 到位置 5（不包括）的字符：

  ```
  b = "Hello, World!"
  print(b[2:5])
  ```

  ```
  llo
  ```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  字符串长度

  实例

  len() 函数返回字符串的长度：

```
a = "Hello, World!"
print(len(a))
```

  字符串方法

  Python 有一组可用于字符串的内置方法。

  实例

  strip() 方法删除开头和结尾的空白字符：

```
a = " Hello, World! "
print(a.strip()) # returns "Hello, World!“
```

  实例

  upper() 方法返回大写的字符串：

```
a = "Hello, World!"
print(a.upper())
```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  实例

  replace() 用另一段字符串来替换字符串：

  ```
  a = "Hello, World!"
  print(a.replace("World", "Kitty"))
  ```

  实例

  split() 方法在找到分隔符的实例时将字符串拆分为子字符串：

  ```
  a = "Hello, World!"
  print(a.split(",")) # returns ['Hello', ' World!’]
  ```

  实例

  检查以下文本中是否存在短语 "ina"：

  ```
  txt = "China is a great country"
  x = "ina" in txt
  print(x)
  ```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  实例

  replace() 用另一段字符串来替换字符串：

  ```
  a = "Hello, World!"
  print(a.replace("World", "Kitty"))
  ```

  ```
  Hello, Kitty!
  ```

  实例

  split() 方法在找到分隔符的实例时将字符串拆分为子字符串：

  ```
  a = "Hello, World!"
  print(a.split(",")) # returns ['Hello', ' World!’]
  ```

  ```
  ['Hello', ' World!']
  ```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  实例

  检查以下文本中是否存在短语 "ina"：

```
txt = "China is a great country"
x = "ina" in txt
print(x)
```

```
True
```

  实例

  检查以下文本中是否没有短语 "ina"：

```
txt = "China is a great country"
x = "ain" not in txt
print(x)
```

```
True
```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  字符串级联（串联）

  如需串联或组合两个字符串，您可以使用 + 运算符。

  实例

  将变量 a 与变量 b 合并到变量 c 中：

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

```
HelloWorld
```

  实例

  在它们之间添加一个空格：

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

```
Hello World
```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  字符串格式

  正如在 Python 变量一章中所学到的，我们不能像这样组合字符串和数字：

  实例

```
age = 63
txt = "My name is Bill, I am " + age
print(txt)
```

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

  但是我们可以使用 format() 方法组合字符串和数字！

  format() 方法接受传递的参数，格式化它们，并将它们放在占位符 {} 所在的字符串中：

  实例

  使用 format() 方法将数字插入字符串：

```
age = 63
txt = "My name is Bill, and I am {}"
print(txt.format(age))
```

```
My name is Bill, and I am 63
```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  字符串格式

  format() 方法接受不限数量的参数，并放在各自的占位符中：

  实例

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

```
I want 5 pieces of item 789 for 24.36 dollars.
```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  字符串格式

  您可以使用索引号 {0} 来确保参数被放在正确的占位符中：

  实例

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

```
I want to pay 49.9 dollars for 3 pieces of item 576.
```

第二章 第一节 Python 基本数据类型

- **Python 字符串**

  字符串方法

  Python 有一组可以在字符串上使用的内建方法。

  注释：所有字符串方法都返回新值。它们不会更改原始字符串。

  https://www.w3school.com.cn/python/python_strings.asp

方法	描述
capitalize()	把首字符转换为大写。
casefold()	把字符串转换为小写。
center()	返回居中的字符串。
count()	返回指定值在字符串中出现的次数。
encode()	返回字符串的编码版本。
endswith()	如果字符串以指定值结尾，则返回 true。
expandtabs()	设置字符串的 tab 尺寸。
find()	在字符串中搜索指定的值并返回它被找到的位置。
format()	格式化字符串中的指定值。
format_map()	格式化字符串中的指定值。
index()	在字符串中搜索指定的值并返回它被找到的位置。
isalnum()	如果字符串中的所有字符都是字母数字，则返回 True。
isalpha()	如果字符串中的所有字符都在字母表中，则返回 True。
isdecimal()	如果字符串中的所有字符都是小数，则返回 True。
isdigit()	如果字符串中的所有字符都是数字，则返回 True。
isidentifier()	如果字符串是标识符，则返回 True。
islower()	如果字符串中的所有字符都是小写，则返回 True。
isnumeric()	如果字符串中的所有字符都是数，则返回 True。
isprintable()	如果字符串中的所有字符都是可打印的，则返回 True。
isspace()	如果字符串中的所有字符都是空白字符，则返回 True。
istitle()	如果字符串遵循标题规则，则返回 True。
isupper()	如果字符串中的所有字符都是大写，则返回 True。
join()	把可迭代对象的元素连接到字符串的末尾。
ljust()	返回字符串的左对齐版本。
lower()	把字符串转换为小写。
lstrip()	返回字符串的左修剪版本。
maketrans()	返回在转换中使用的转换表。
partition()	返回元组，其中的字符串被分为三部分。
replace()	返回字符串，其中指定的值被替换为指定的值。
rfind()	在字符串中搜索指定的值，并返回它被找到的最后位置。
rindex()	在字符串中搜索指定的值，并返回它被找到的最后位置。
rjust()	返回字符串的右对齐版本。
rpartition()	返回元组，其中字符串分为三部分。
rsplit()	在指定的分隔符处拆分字符串，并返回列表。
rstrip()	返回字符串的右边修剪版本。
split()	在指定的分隔符处拆分字符串，并返回列表。
splitlines()	在换行符处拆分字符串并返回列表。
startswith()	如果以指定值开头的字符串，则返回 true。
strip()	返回字符串的剪裁版本。
swapcase()	切换大小写，小写成为大写，反之亦然。
title()	把每个单词的首字符转换为大写。
translate()	返回被转换的字符串。
upper()	把字符串转换为大写。
zfill()	在字符串的开头填充指定数量的 0 值。

第二章 第一节 Python 基本数据类型

- **Python 布尔**

  布尔表示两值之一：True 或 False。

  布尔值

  在编程中，您通常需要知道表达式是 True 还是 False。

  您可以计算 Python 中的任何表达式，并获得两个答案之一，即 True 或 False。

  比较两个值时，将对表达式求值，Python 返回布尔值答案：

  实例

```
print(8 > 7)
print(8 == 7)
print(8 < 7)
```

```
True
False
False
```

第二章 第一节 Python 基本数据类型

- **Python 布尔**

  当在 if 语句中运行条件时，Python 返回 True 或 False：

  实例

  根据条件是对还是错，打印一条消息：

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

```
b is not greater than a
```

第二章 第一节 Python 基本数据类型

- **Python 布尔**

  评估值和变量

  bool() 函数可让您评估任何值，并为您返回 True 或 False。

  实例

  评估字符串和数字：

```
print(bool("Hello"))
print(bool(10))
```

```
True
True
```

  实例

  评估两个变量：

```
x = "Hello"
y = 10
print(bool(x))
print(bool(y))
```

```
True
True
```

第二章 第一节 Python 基本数据类型

- **Python 布尔**

  评估值和变量

  大多数值都为 True

  如果有某种内容，则几乎所有值都将评估为 True。

  除空字符串外，任何字符串均为 True。

  除 0 外，任何数字均为 True。

  除空列表外，任何列表、元组、集合和字典均为 True。

  实例

  下例将返回 True：

```
bool("abc")
bool(123)
bool(["apple", "cherry", "banana"])
```

```
True
True
True
```

第二章 第一节 Python 基本数据类型

- **Python 布尔**

  某些值为 False

  实际上，除空值（例如 ()、[]、{}、""、数字 0 和值 None）外，没有多少值会被评估为 False。当然，值 False 的计算结果为 False。

  实例

  下例会返回 False：

  ```
  bool(False)
  bool(None)
  bool(0)
  bool("")
  bool(())
  bool([])
  bool({})
  ```

第二章 第一节 Python 基本数据类型

- **Python 布尔**

  某些值为 False

  当在 if 语句中运行条件时，Python 返回 True 或 False：

  在这种情况下，一个值或对象的计算结果为 False，即如果对象由带有 __len__ 函数的类生成的，且该函数返回 0 或 False：

  实例

```
class myclass():
  def __len__(self):
    return 0
myobj = myclass()
print(bool(myobj))
```

第二章 第一节 Python 基本数据类型

- **Python 布尔**

  函数可返回布尔

  Python 还有很多返回布尔值的内置函数，例如 isinstance() 函数，该函数可用于确定对象是否具有某种数据类型：

  实例

  检查对象是否是整数：

  ```
  x = 200
  print(isinstance(x, int))
  ```

  ```
  True
  ```