Data Type

Python 基础入门

回顾

常量 和 变量

变量的 命名 规则

保留名

赋值 操作

数值 运算

表达式

变量命名

好的 变量命名 习惯

- ✓ 使用 有意义 的名称以提高 代码可读性
- ✓ 使用 下划线 分割单词：student_number_of_the_class
- ✓ 使用 驼峰命名法：PlayStation（大写）、iPhone（小写）
- ✓ 选择你喜欢的命名习惯，保持 一致 即可

类型

常量 和 变量 都有对应的 类型

- ✓ 常量的类型 固定不变
- ✓ 变量的类型 可变，取决于所存的值

类型 不同，Python 提供的 操作 也不同

- ✓ 对两个数字而言，加号 表示 加法
- ✓ 对两个字符串而言，加号 表示 拼接

```
>>> my_int = 1 + 4
>>> print(my_int)
5
>>> my_str = "Hello" + "World!"
>>> print(my_str)
HelloWorld!
>>>
```

类型

类型 很重要!

Python 规定了每种类型的 使用方法

有些操作是 无法执行 的

```
>>> my_str = "Hello" + "World!"
>>> my_str = my_str + 1
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    my_str = my_str + 1
TypeError: must be str, not int
```

获取类型

使用 type() 获取常量和变量的类型

```
>>> type(my_str)
<class 'str'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
```

```
>>> type('int')
<class 'str'>
>>> type("float")
<class 'str'>
>>> type('''
str
''')
<class 'str'>
>>> type("""
str
""")
<class 'str'>
```

Integer

Float

String

类型转换

Python 会执行一些默认的 类型转换

- ✓ 表达式中既有 int 又有 float 时，结果为 float
- ✓ 两个整数之间的除法，结果为 float

可以使用 int() 和 float() 进行类型转换

和 print() 类似的用法（函数）

```
>>> type(1 * 5)
<class 'int'>
>>> type(1.0 * 5)
<class 'float'>
```

```
>>> x = 10 / 2
>>> type(x)
<class 'float'>
```

```
>>> x = int(x)
>>> type(x)
<class 'int'>
>>> x = float(x)
>>> type(x)
<class 'float'>
```

类型转换

数字 和 字符串 之间的类型转换

使用 int()、float()、str()

将类型转换为 int、float 和 string

字符串中包含 非数字字符 时，转换将 出错

```
>>> my_str = '123'
>>> type(my_str)
<class 'str'>
>>> print(my_str + 1)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print(my_str + 1)
TypeError: must be str, not int
>>> my_str = int(my_str)
>>> type(my_str)
<class 'int'>
>>> print(my_str + 1)
124
```

```
>>> my_str = 'Hello World!'
>>> my_str = int(my_str)
Traceback (most recent call last):
  File "<pyshell#33>", line 1, in <module>
    my_str = int(my_str)
ValueError: invalid literal for int() with base 10: 'Hello World!'
```

类型转换

转换之后，有时需要保留 原始数据

```
>>> my_str = '123'
>>> my_str = int(my_str)
>>> print(my_str + 1)
124
```

```
>>> my_str = '123'
>>> my_int = int(my_str)
>>> print(my_int + 1)
124
```

```
>>> my_str = '123'
>>> print(int(my_str) + 1)
124
```

用户输入

使用 input() 输入信息

- ✓ Python 会 停下来
- ✓ 等待用户 输入 并 回车

输入的内容会存到一个 字符串 中

```
>>> name = input('Who are you?\n')
Who are you?
Honlan
>>> print('Welcome', name)
Welcome Honlan
```

```
>>> my_int = input("Input a number:")
Input a number:7
>>> type(my_int)
<class 'str'>
>>>
```

用户输入

将用户输入转换为 数字 并计算

```
>>> weight = float(input('Weight of apple:'))
Weight of apple:0.6
>>> price = float(input('Price of apple:'))
Price of apple:12
>>> cost = weight * price
>>> print(cost)
7.199999999999999
```

格式化

控制小数点的 位数

```
>>> weight = float(input('Weight of apple:'))
Weight of apple:0.6
>>> price = float(input('Price of apple:'))
Price of apple:12
>>> cost = weight * price
>>> print(cost)
7.199999999999999
```

```
>>> print('%.3f' % cost)
7.200
>>> print('%.2f' % cost)
7.20
>>> print('%.1f' % cost)
7.2
>>> print('%.0f' % cost)
7
```

练习

投资

- ✓ 有一笔钱用于投资，金额用 money 表示，单位为元
- ✓ 投资期限用 month 表示，单位为月
- ✓ 每个月的利息用 rate 表示，单位为百分之一
- ✓ 按每个月计算复利，那么到期后可以得到多少 利息 ？
- ✓ 使用 input() 交互地输入以上三项数据
- ✓ 使用 Python脚本 存储代码

注释

以 # 开头的代码行为 注释

- ✓ 可以在注释中写一些 说明信息
- ✓ 被注释的代码行 不会运行
- ✓ 可以把注释理解为 关掉 代码
- ✓ 使用 三引号 进行多行注释（在 Python脚本 中）

```
>>> # A Wonderful Code is Coming!
>>> # Written by Honlan, 2017-09-21
>>> print('A Wonderful Code')
A Wonderful Code
```

```
>>> print('Print this message')
Print this message
>>> # print('This message will not be print')
```

作业

1 翻转 一个浮点数

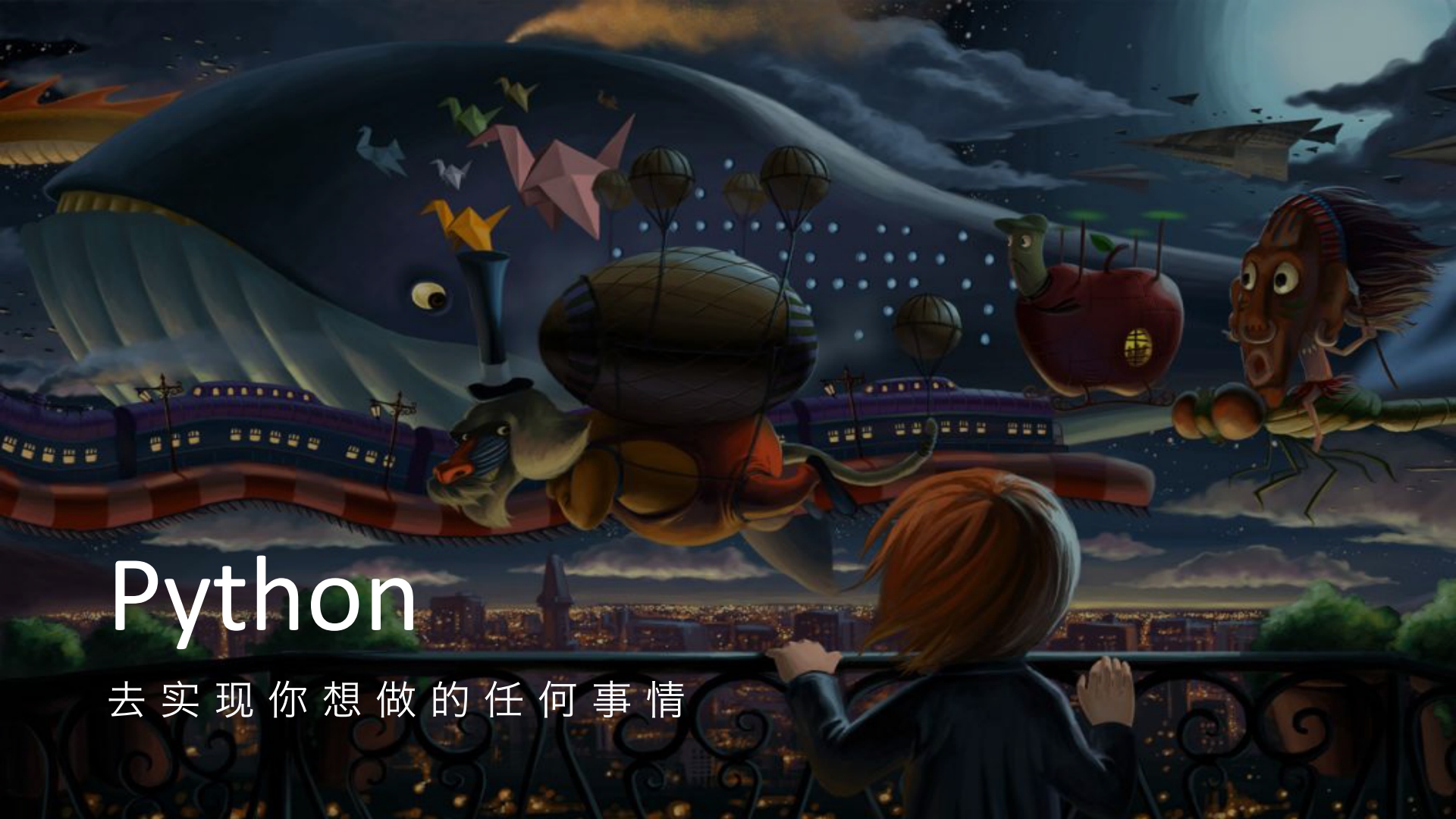
- ✓ 例如，将 123.456 翻转为 654.321
- ✓ 使用 input()，从而可以由用户输入浮点数

```
>>> my_str = 'Hello World!'
>>> my_str = my_str[::-1]
>>> print(my_str)
!dlroW olleH
```

作业

2 圆 的周长和面积

- ✓ 使用 input() 输入圆的 半径
- ✓ 计算并打印圆的 周长 和 面积


Python
去 实 现 你 想 做 的 任 何 事 情