1 变量与类型

1.1 变量

1.1.1 变量的概念

在 Python 中，变量是用来存储数据的标识符。它就像一个容器，可以存放各种数据，如数字、文本、列表等。

变量的命名规则：

- 由字母、数字和下划线组成，且不能以数字开头
- 不能使用 Python 的保留字（如 if、else、for 等）
- 变量名区分大小写

1.1.2 变量的赋值

```
# 变量赋值示例
age = 25
name = "张三"
is_student = True

# 打印变量值
print(age)        # 输出：25
print(name)       # 输出：张三
print(is_student) # 输出：True
```

1.2 基本数据类型

1.2.1 数值类型

数值类型包括整数 (int)、浮点数 (float) 和复数 (complex)。

```
# 整数类型
num1 = 10
num2 = -20
print(type(num1))  # <class 'int'>

# 浮点数类型
price = 19.99
temperature = -3.5
print(type(price))  # <class 'float'>

# 复数类型
complex_num = 3 + 4j
print(type(complex_num))  # <class 'complex'>
```

1.2.2 布尔类型

布尔类型只有两个值：True 和 False。

```
is_valid = True
has_error = False

print(is_valid)    # True
print(has_error)   # False
print(type(is_valid))  # <class 'bool'>
```

1.2.3 字符串类型

字符串是由字符组成的序列，使用单引号、双引号或三引号定义。

```
# 单引号字符串
name = 'Python'
print(name)  # Python

# 双引号字符串
greeting = "Hello, world!"
print(greeting)  # Hello, world!

# 三引号多行字符串
poem = '''This is a poem
with multiple lines.
It's perfect for long texts.'''
print(poem)
```

1.2.4 列表类型

列表是有序的、可变的集合，可以包含不同类型的数据。

```
# 创建列表
numbers = [1, 2, 3, 4, 5]
mixed_list = [1, "hello", 3.14, True]

# 访问列表元素
print(numbers[0])   # 1
print(mixed_list[2]) # 3.14

# 列表操作
numbers.append(6)   # 添加元素
print(numbers)      # [1, 2, 3, 4, 5, 6]

numbers.remove(3)   # 删除元素
```

```
print(numbers)      # [1, 2, 4, 5, 6]
```

1.2.5 元组类型

元组与列表类似，但不可变（一旦创建不能修改）。

```
# 创建元组
coordinates = (10, 20)
days = ("Monday", "Tuesday", "Wednesday")

# 访问元组元素
print(coordinates[0])  # 10

# 元组不能修改
# days[0] = "Sunday"  # 会报错
```

1.2.6 集合类型

集合是无序的、不重复的元素集合。

```
# 创建集合
fruits = {"apple", "banana", "orange"}

# 集合操作
fruits.add("grape")     # 添加元素
print(fruits)           # {'apple', 'banana', 'orange', 'grape'}

fruits.remove("banana") # 删除元素
print(fruits)           # {'apple', 'orange', 'grape'}
```

1.2.7 字典类型

字典是键值对的集合，用于存储关联数据。

```
# 创建字典
person = {
    "name": "张三",
    "age": 25,
    "is_student": True
}

# 访问字典元素
print(person["name"])       # 张三
print(person.get("age"))    # 25

```

```python
# 修改字典元素
person["age"] = 26
print(person)               # {'name': '张三', 'age': 26, 'is_student': True}

# 添加新元素
person["email"] = "zhangsan@example.com"
print(person)
```

1.3 数据类型转换

Python 允许在不同数据类型之间进行转换。

```python
# 整数转浮点数
num = 10
print(float(num))  # 10.0

# 浮点数转整数
price = 9.99
print(int(price))  # 9

# 数字转字符串
count = 5
print(str(count))  # '5'

# 字符串转整数
text_num = "123"
print(int(text_num))  # 123

# 字符串转浮点数
text_price = "9.99"
print(float(text_price))  # 9.99
```

1.4 常用字符串操作

字符串是 Python 中最常用的数据类型之一，这里介绍一些常用操作：

```python
# 字符串长度
text = "Hello, Python!"
print(len(text))  # 13

# 字符串拼接
greeting = "Hello"
name = "World"
message = greeting + ", " + name + "!"
```

```python
print(message)  # Hello, World!

# 字符串格式化
name = "Alice"
age = 25
print(f"My name is {name} and I'm {age} years old.")

# 字符串切片
text = "Python is fun!"
print(text[0:6])   # Python
print(text[7:9])   # is
print(text[-3:])   # fun
```

1.5 示例代码

以下是一个完整的示例，展示如何使用不同数据类型：

```python
# 变量和数据类型示例

# 定义变量
name = "张三"           # 字符串
age = 25                # 整数
height = 175.5          # 浮点数
is_student = True       # 布尔值

# 定义集合
hobbies = {"reading", "hiking", "gaming"}

# 定义字典
person = {
    "name": name,
    "age": age,
    "height": height,
    "is_student": is_student,
    "hobbies": hobbies
}

# 打印输出
print(f"姓名: {person['name']}")
print(f"年龄: {person['age']}")
print(f"身高: {person['height']}cm")
print(f"是否学生: {person['is_student']}")
print(f"爱好: {person['hobbies']}")

```

```python
# 字符串操作示例
message = "Python is a great programming language!"
print(f"消息长度: {len(message)}")
print(f"消息大写: {message.upper()}")
print(f"消息分割: {message.split()}")
```

通过以上内容，对 Python 中的变量和基本数据类型有了基本了解。在实际编程中，这些基本概念是构建复杂程序的基础。

1.6 保留字（关键字）

Python 的保留字（也称为关键字）是语言中具有特殊含义的预定义标识符，它们不能用作变量名、函数名或其他标识符。Python 的保留字数量相对较少，这使得 Python 的语法简洁而清晰。

以下是 Python 3.x 版本中的保留字列表：

保留字	保留字	保留字	保留字
False	None	True	and
as	assert	async	await
break	class	continue	def
del	elif	else	except
finally	for	from	global
if	import	in	is
lambda	nonlocal	not	or
pass	raise	return	try
while	with	yield	

这些保留字在 Python 中有特殊的语法功能，用于控制程序的流程、定义数据结构、进行逻辑运算等。例如：

- if、elif、else 用于条件语句
- for、while 用于循环结构
- def 用于定义函数
- class 用于定义类
- try、except、finally 用于异常处理

由于保留字的特殊性，你不能将它们用作标识符（如变量名、函数名等），否则会导致语法错误。例如，下面的代码会导致错误：

```python
# 错误示例：使用保留字作为变量名
return = 5  # SyntaxError: invalid syntax
```

如果你不确定某个词是否是保留字，可以使用 Python 的内置模块 keyword 来查询：

```
import keyword

for kw in keyword.kwlist:
    print(kw)
```

这段代码会输出当前 Python 版本中的所有保留字列表。

输出的保留字列表如下：

```
False
None
True
and
as
assert
async
await
break
class
continue
def
del
elif
else
except
finally
for
from
global
if
import
in
is
lambda
nonlocal
not
or
pass
raise
return
try
while
with
yield
```