

# 实验一：Python变量、简单数据类型

班级 24计科1

学号 B2024000001

姓名 张三

大纲：

- Python变量
- 字符串
- 数: 整数、浮点数以及数学运算
- Python之禅: Python编程哲学

## 实验注意事项

1. 请在指定的地方按照实验指导要求来编写代码。
2. 请按照实验指导要求使用指定的变量名或函数名，不要使用其他的名字。
3. 不要添加任何额外的语句。
4. 不要添加任何额外的代码单元格。
5. 不要在不需要的地方修改作业代码，比如创建额外的变量，修改测试文件中的代码。
6. 实验指导中的 ... 表示需要你补充代码的部分，其他部分的代码不用修改。
7. 代码提示中会给出估计的代码行数，例如大约1行代码，估计的代码行数只是一个参考值，实际编写时可能会有出入，请根据实际情况来编写。
8. 请独立完成作业，禁止抄袭，发现抄袭行为成绩记零分。

```
In [1]: # 导入测试代码
from testset1 import *
```

## Python变量

Python变量不用指定类型，解释器会自动推断变量的数据类型

```
In [2]: message = 'Hello World!' # 字符串类型 (str)
numbers = 100 # 整数类型 (int)

print(message)
print(numbers)
```

```
Hello World!
100
```

## 习题一

请在下面的代码单元格中定义两个变量 `name` 和 `student_id`，并分别赋值为你的名字和学号，注意请将你的姓名和学号放在双引号或者单引号中，例如 `name = '张三'`

```
In [3]: # student_name = '...' # 请在此处输入你的名字
# student_id = '...' # 请在此处输入你的学号
# 大约两行代码
# 你编写的代码从这里开始
student_name = '...'
```

```
student_id = '...'

# 你编写的代码到这里结束
```

```
In [4]: test_name_and_student_id(student_name, student_id)
```

你的名字是: ...  
你的学号是: ...  
恭喜你通过了习题一: test\_name\_and\_student\_id测试。1/7

变量随时可以再被赋予任意类型的值:

```
In [5]: message = 3.14
numbers = 'Hello World!'

print(message)
print(numbers)
```

```
3.14
Hello World!
```

变量不是盒子，而是标签

 vars are not boxes, they are tags

盒子和标签关键区别:


1. 变量只是对象的引用（类似于C语言中的指针），而不是存储对象的容器
2. 赋值操作是把标签贴到对象上，而不是把值复制到盒子里
3. 多个变量可以引用同一个对象
4. 对象的生命周期由引用计数决定，而不是作用域

```
In [6]: a = 10      # 变量a是一个标签，贴在整数对象10上
b = a        # 创建变量b，也贴在和变量a一样指向的整数对象10上
a = 20       # a贴到整数对象20上，b仍然贴在10上，变量不是盒子，而是标签
print(b)     # 输出10
```

```
10
```

变量命名规则:

- 使用英语单词(蛇形命名法): my\_message, first\_name
- 变量名称由数字、字母(包括大写字母和小写字母)、下划线组成。
- 变量名不能以数字开头
- 变量名不能用python关键字和内置函数的名字（见下图）
- 变量命名严格区分大小写

 keywords and builtin functions

## 习题二

关于下面这段代码，ABCD哪一个选项是正确的？

```
value = 100
msg = 'hello'
value = msg
```

A. value的值是100，msg的值是'hello'

B. value的值是'hello'，msg的值是'hello'

C. value的值是'hello'，msg的值是100

D. value的值是100，msg的值是100

```
In [7]: # answer2 = ...           # 请在此处输入你的答案,例如: answer2 = "A"
# 大约一行代码
# 你编写的代码从这里开始
answer2 = "... "

# 你编写的代码到这里结束
```

```
In [ ]: test_quiz2(answer2)
```

## 函数

函数是Python程序中最常见的代码单元，我们的要完成的实验以及实验的测试代码都是通过函数来实现的。

```
In [ ]: # 定义函数
def greet_user():
    """Display a simple greeting."""
    print("Hello!")
    print("欢迎参加Python课程！")
```

```
In [ ]: # 调用函数
greet_user()
```

- **def** 表示要定义一个函数
- **greet\_user()** 是定义的函数名，必须要有圆括号
- 函数名同样要遵守Python变量的命名规则
- 函数名后面的后面是冒号，表示函数代码块开始，代码块可以包含任意数量的代码行
- 函数体的代码块每一行都必须有缩进
- 按照代码规范，第一句是函数的文档注释(docstring), 用三引号括起来

```
In [ ]: # 定义一个带参数，有返回值的函数
def subtract_x_and_y(x, y):
    """Return x - y."""
    z = x - y
    return z
```

```
In [ ]: # 调用函数并打印返回值
print(subtract_x_and_y(3, 2))
```

- 在 **def subtract\_x\_and\_y(x, y):** 中，x和y是形参，函数定义时的参数
- 在函数中使用 **return** 语句返回值
- **subtract\_x\_and\_y(3, 2)** 调用函数时, 3是x的实参，2是y的实参，3和2被称为位置参数，位置参数的数量和顺序都不能错

## 习题三

```
In [ ]: # 编写一个函数subtract_abc, 包括三个位置参数a,b,c, 计算a-b-c的值, 并返回结果
# def your_function(参数列表):
#     """函数文档字符串"""
#     result = ...
#     return result
# 大约四行代码

# 你编写的代码从这里开始
def your_function(参数列表):
    """函数文档字符串"""
    result = ...
    return result

# 你编写的代码到这里结束
```

```
In [ ]: test_subtract_abc(subtract_abc)
```

## 字符串（一）

字符串就是一系列字符，使用单引号或者双引号扩起来

```
In [ ]: message = 'hello'
name = "ada lovelace"
```

Python语言中没有区分字符（char）和字符串（str），字符就是长度为1的字符串

字符串的方法：

- title方法：将单词的首字母大写
- 将字符串改为全部大写或者小写: upper(), lower()
- format方法或者f字符串：字符串的格式化
- 字符串中的转义字符：制表符'\t', 换行'\n'
- 删除空白: rstrip(), lstrip(), strip()

将单词的首字母大写

```
In [ ]: name.title()
```

将字符串改为全部大写或者小写: upper(), lower()

```
In [ ]: print(name.upper())
print(name.lower())
```

f 字符串(Python 3.6+)

- 可以引用 {变量}

```
In [ ]: first_name = 'ada'
last_name = 'lovelace'
full_name = f'The first programmer is {first_name.title()} {last_name.title()}' # f-->
print(full_name)
```

format方法

```
In [ ]: full_name = '{} {}'.format(first_name, last_name)
```

```
print(full_name)
```

## 习题四

补充完成下面的函数，函数的输入是字符串 `first_name` 和 `last_name`，函数返回值是向用户问候的信息，例如输入 `first_name='albert'`，`last_name='einstein'`，返回值是 `'Hello, Albert Einstein!'`

```
In [ ]: def greet(first_name, last_name):
        # greeting = f"..."      # 请使用f-string格式化字符串
        # 大约一行代码
        # 你编写的代码从这里开始
        greeting = f"..."

        # 你编写的代码到这里结束
        return greeting
```

```
In [ ]: # 习题四的测试代码
        test_greet(greet)
```

## 字符串（二）

使用制表符或换行符添加空白

```
In [ ]: print('Language:\n\tPython\n\tC\n\tJavaScript')
```

删除空白

```
In [ ]: favorite_language = ' python '
        print(favorite_language)
```

```
In [ ]: favorite_language.rstrip()
```

```
In [ ]: favorite_language = 'pyt      hon '
        favorite_language.rstrip()
```

```
In [ ]: favorite_language.lstrip()
```

```
In [ ]: favorite_language.strip()
```

避免字符串的语法错误, 如果字符串中出现了单引号，可以使用双引号括起来

```
In [ ]: # message = 'One of Python's strengths is its diverse community.'
```

```
In [ ]: message = "One of Python's strengths is its diverse community."
        print(message)
```

也可以使用转义字符（Escape Character）来显示一些特殊的字符，Python主要的转义字符包括：

- `\n` - 换行

- `\t` - 制表符(Tab)
- `\\` - 反斜杠
- `\'` - 单引号
- `\"` - 双引号
- `\r` - 回车
- `\b` - 退格

```
In [ ]: message = "One of Python\'s strengths is its diverse community."
        print(message)
```

## 数与数学运算（一）

- 整数（`int`）：没有区分长度（没有`int32`，`int64`，`long`），从Python 3.8开始没有最大值的限制
- 浮点数(`float`): 没有区分单精度和双精度
- `int` 和 `float` 的实际长度会根据机器平台来决定，绝大多数情况下为64位，8个字节
- 这里讲解的所有的运算都可以使用整数和浮点数来进行

基本运算加减乘除所使用的运算符：`+`, `-`, `*`, `/`

```
In [ ]: 2 + 3
```

```
In [ ]: 3 - 2
```

```
In [ ]: 3 * 2.5 # 有一个操作数为浮点数,结果一定是浮点数
```

```
In [ ]: 3 / 1 # 结果一定是浮点数
```

乘方运算:`**`

```
In [ ]: 3 ** 3
```

```
In [ ]: 2.25 ** .5 # 指数也可以是小数, 指数0.5表示开平方
```

```
In [ ]: 0 ** 0 # 比较特殊的运算结果: 0的0次方是1
```

模运算：`%`（得到余数）

```
In [ ]: 5 % 3
```

`%1` 可以获取一个浮点数的小数部分

```
In [ ]: 67.14 % 1 # 浮点数的运算不是完全精确的
```

```
In [ ]: 3.14159 % .1
```

除法求商：`//`

```
In [ ]: 8 // 3
```

// 1 可以去掉一个浮点数的小数部分

```
In [ ]: 5.25 // 1
```

求商和余数: `divmod` 函数

```
In [ ]: divmod(20, 3)
```

`round` 函数: 浮点数四舍五入

```
In [ ]: round(.6666)
```

```
In [ ]: round(1.2)
```

```
In [ ]: round(.50001)
```

虚数的运算

```
In [ ]: (-1) ** 0.5
```

```
In [ ]: x = 3 + 2j    # 复数的表示方法: 实部+虚部 , j表示虚数单位
print(x)             # 打印复数
print(x.real)         # 打印实部
print(x.imag)         # 打印虚部
```

## 习题五

已知函数的输入是直角三角形的两个直角边的长度, 函数的返回值是直角三角形的斜边(hypotenuse)的长度, 使用勾股定理计算斜边的长度

```
In [ ]: def get_hypotenuse(a, b):
        # c = ...          # 请在此处输入你的答案
        # 大约一行代码
        # 你编写的代码从这里开始
        c = ...

        # 你编写的代码到这里结束
        return c
```

```
In [ ]: test_get_hypotenuse(get_hypotenuse)
```

更多的数学运算函数, 可以通过导入 `math` 模块来使用, 例如要计算 `sin` 三角函数的代码如下:

```
In [ ]: import math

print(math.sin(math.pi / 2))    # sin(90°) = 1
```

更多的 `math` 模块的数学运算函数请查看[math模块文档](#)

## 习题六

已知三角形的边a, b 以及他们的夹角theta弧度, 计算第三条边c的长度, 使用余弦定理计算第三条边的长度

$$c = \sqrt{a^2 + b^2 - 2ab \cos(\theta)}$$

```
In [ ]: import math

def get_third_side(a, b, theta_radians):
    # c = ...          # 使用余弦公式计算第三条边的长度
    # 大约1行代码
    # 你编写的代码从这里开始
    c = ...
    # 你编写的代码到这里结束
    return c
```

```
In [ ]: test_get_third_side(get_third_side)
```

## 习题七

求离整数n最近的平方数

例如, 如果n=111, 那么nearest\_sq(n)等于121, 因为111比100 (10的平方) 更接近121 (11的平方)。

如果n已经是完全平方 (例如n=144, n=81, 等等), 你需要直接返回n。

```
In [ ]: def nearest_sq(n):
    # result = ...          # 求离开n最近的完全平方数
    # 大约1到2行代码
    # 你编写的代码从这里开始
    result = ...

    # 你编写的代码到这里结束
    return result
```

```
In [ ]: test_nearest_sq(nearest_sq)
```

## 数与数学运算 (二)

为什么浮点数运算不精确相等, 例如0.3+0.1+0.2不等于0.6?

```
In [ ]: .3 + .1 + .2 == .6
```

打印0.3, 0.2, 0.1 的近似值, 精确到小数点后20位。

```
In [ ]: print(f'{.3:.20f}')
print(f'{.2:.20f}')
print(f'{.1:.20f}')
print(f'{.6:.20f}')
```

为什么浮点数运算不精确相等, 例如0.3+0.1+0.2不等于0.6?



因为所有的浮点数（十进制）在计算机中都是以二进制的形式存储的，而二进制浮点数无法精确表示大部分的十进制小数，例如0.1在二进制中是一个无限循环小数，所以在计算机中0.1的存储是一个近似值。

数字中的下划线

```
In [ ]: universe_age = 14_000_000_000
universe_age
```

科学计数法

```
In [ ]: universe_age = 1.4e10
universe_age
```

同时给多个变量赋值

```
In [ ]: x, y, z = 0, 0, 0
x
```

常量： 常量名应该全部大写

```
In [ ]: MAX_CONNECTIONS = 5000
```

Python语法没有强制约定常量不能被修改

```
In [ ]: MAX_CONNECTIONS = 15000
```

## 字符串和数字之间的转化

- `str()`函数：将其他数据转化为字符串
- `int()`函数：将其他数据转化为整数
- `float()`函数：将其他数据转化为浮点数

```
In [ ]: str1 = str(123)
str1
```

```
In [ ]: int1 = int('123')
int1
```

```
In [ ]: # 下面的代码会出错，因为'123.4'不是一个整数
# int2 = int('123.4')
# int2
```

```
In [ ]: f1 = float('123.4')
f1
```

```
In [ ]: f2 = float('123')
f2
```

三个关于变量的函数

- `type`函数: 返回该变量的类型
- `id`函数: 返回该变量的id, 这是一个int类型的值
- `isinstance`函数: 如果该变量是某类型的实例, 返回True, 否则返回False

```
In [ ]: message = "Hello"
        number = 42
        pi = 3.14159
```

```
In [ ]: print(id(message), id(number), id(pi), sep=", ")
```

```
In [ ]: print(type(message), type(number), type(pi), sep=', ')
```

```
In [ ]: print(isinstance(message, str), isinstance(number, int), isinstance(pi, float), sep=', ')
```

## 简单变量的值是不可变的

简单类型变量的值（`str,int,float`）都是不可变的, 如果进行改变简单变量的操作, 实际上是创建了新的变量（变量不是盒子, 而是标签）

```
In [ ]: number = 42
        print(id(number))
        number = 100
        print(id(number))
```

数字 `42` 这个值在内存堆中是不能被改变的, 它的id就是保存值的内存的地址, 如果没有变量引用这个id的值, 内存堆中的值将被垃圾回收。

## is 和 ==

`is` 和 `==` 的区别:

- `is` 比较的是两个变量的id是否相等
- `==` 比较的是两个变量的值是否相等

运行下面的代码, 思考为什么会这样? 总结一下在哪些情况下两个不同的变量会有相同的id?

```
In [ ]: a = 256
        b = 256
        print(id(a), id(b))
        print(a is b)
```

```
In [ ]: a = 257
        b = 257
        print(id(a), id(b))
        print(a is b)
```

```
In [ ]: a = -5
        b = -5
        print(id(a), id(b))
        print(a is b)
```

```
In [ ]: a = -6
        b = -6
        print(id(a), id(b))
        print(a is b)
```

从-5到256的整数作为被常用的整数会被缓存起来，这些整数的id总是相同的

```
In [ ]: # 简单字符的字符串也会被缓存
        a = 'helloworld123'
        b = 'helloworld123'
        print(id(a), id(b))
        print(a is b)
```

```
In [ ]: # 因为字符串中有空格，这不是一个简单字符串，不会被缓存
        a = 'hello world'
        b = 'hello world'
        print(id(a), id(b))
        print(a is b)
```

```
In [ ]: a = 'hello'
        b = 'he'
        b += 'llo'
        print(id(a), id(b))
        print(a is b)
```

```
In [ ]: a = tuple()
        b = tuple()
        print(id(a), id(b))
        print(a is b)
```

## Python之禅

```
In [ ]: import this
```

As Abelson and Sussman wrote in the preface to Structure and Interpretation of Computer Programs (MIT Press, 1996),

Programs must be written for people to read, and only incidentally for machines to execute.

## 实验代码自动评分

请运行下面的代码进行自动评分

```
In [ ]: grade_all_tests([(student_name, student_id), answer2,
                          subtract_abc, greet, get_hypotenuse,
                          get_third_side, nearest_sq])
```

## 实验总结

请尽可能使用自己的语言来回答下面的问题。

### 问题1

总结一下Python变量的命名规则以及命名规范。请举一个好的变量命名的例子。

## 问题2

简单变量(int, float, str)的值是不可变的。请解释如果对一个整数进行运算时，为什么整数变量的值不会改变？如何知道简单变量的值是不可变的？

## 生成实验报告

In [ ]:

```
import sys
import os

# Add the root directory (parent directory of `util`) to sys.path
root_dir = os.path.abspath("../") # Adjust based on the structure
sys.path.append(root_dir)

from util import notebook2pdf # Use absolute import after changing sys.path

notebook_file = "实验一-Python变量.ipynb"
html_file = "notebook.html"
pdf_file = "output.pdf"

notebook2pdf.notebook_to_html(notebook_file, html_file)
notebook2pdf.html_to_pdf(html_file, pdf_file)
```

Notebook successfully converted to HTML: notebook.html  
Successfully converted notebook.html to output.pdf

In [ ]:

```
import sys
import os

# Add the root directory (parent directory of `util`) to sys.path
root_dir = os.path.abspath("../") # Adjust based on the structure
sys.path.append(root_dir)

from util import notebook2pdf # Use absolute import after changing sys.path

notebook_file = "实验一-Python变量.ipynb"
html_file = "notebook.html"
pdf_file = "output.pdf"

notebook2pdf.convert_notebook_to_webpdf(notebook_file, pdf_file)
```