第 1 次课 公式的计算

Python 科学计算

周吕文

宁波大学，机械工程与力学学院

2024 年 9 月 1 日

提要

竖直上抛运动

竖直上抛运动公式

$$y = v_0 t - \frac{1}{2} g t^2$$

如果 $v_0 = 5\ \mathrm{m/s}$，$g = 9.8\ \mathrm{m/s^2}$，$t = 0.6\ \mathrm{s}$，求 y

$$y = v_0 t - \frac{1}{2} g t^2 = 5 - 0.5 \times 9.8 \times 0.6^2 = 3.236\ \mathrm{m}$$

如果 $v_0 = 5\ \mathrm{m/s}$，$g = 9.8\ \mathrm{m/s^2}$，$y = 0$，求 t

$$v_0 t - \frac{1}{2} g t^2 = 0 \quad \Longrightarrow \quad t_1 = 0\,\mathrm{s} \ \text{或} \ t_2 = \frac{2 v_0}{g} = \frac{2 \times 5}{9.8} = 1.02\,\mathrm{s}$$

计算器与使用变量

用程序作计算器

```
>>> 5*0.6 - 0.5*9.8*0.6**2
1.236
>>> 2*5/9.8
1.0204081632653061
```

使用变量

ball.py

```
v0 = 5
g = 9.8
t = 0.6
y = v0*t - 1/2 * g*t**2
t2 = 2*v0/g
print(y)
print(t2)
```

```
>>>
1.236
1.0204081632653061
```

变量名

变量命名规则

- 变量名只能包含字母、数字和下划线。
- 变量名不能以数字开头。
- 变量名不能包含空格，但可以使用下划线来分隔单词。

变量命名尽量与问题的数学描述保持一致，兼顾描述性和简洁性

```
initial_velocity = 5
acceleration_of_gravity = 9.8
TIME = 0.6
VerticalPositionOfBall = initial_velocity * TIME - \
                0.5 * acceleration_of_gravity * TIME**2
print(VerticalPositionOfBall)
```

保留字

python 有 33 个保留字

and	as	assert	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	True	False	None

不要用保留字作为变量名和文件名

```
>>> lambda = 5
  File "<stdin>", line 1
    lambda = 5
           ^
SyntaxError: invalid syntax
>>> lambda_ = 5
>>> print(lambda_)
5
```

注释

注释的内容在程序执行时都会被忽略

- 对程序调试、阅读、共同开发和后续使用、修改都具有重要价值
- 提供重要但代码不能明确体现的信息，勿重复含义很清楚的内容

注释的两种方式

ball_comments.py

```
'''
要同时注释多行可以像这样用三个引号引起来（单引和双引都可以）
这是计算竖直上抛运动过程中小球位置的程序
'''
# 如果要注释单行，可以像这样直接在行首加“#”
v0 = 5                          # 初速度
g = 9.8                         # 重力加速度
t = 0.6                         # 时间
y = v0*t - 1/2 * g*t**2         # 位置
print(y)
```

指定文字和数字的输出格式：printf 风格

输出结果时可能需要包含特定格式的文字和数字

```
t = 0.6
y = 1.2342
print('At t = %g s, y is %.2f m' % (t, y))
```

```
>>>
At t = 0.6 s, y is 1.23 m
```

语法解释

- 占位符以一个百分号开始，后面跟着占位符格式说明
- 字符串后跟一个百分号和一个圆括号，括号内按顺序列出对应变量
- %g ← t 最短十进制实数，%.2f ← y 精确到小数点后两位的实数

指定文字和数字的输出格式：printf 风格

常用的 printf 风格的格式规范

%s	字符串
%d	整数
%0xd	整数，其字段宽度为 x，不足则在左侧以 0 填补
%f	6 位小数的十进制记数法
%e	紧凑的科学记数法，指数用 e 表示
%E	紧凑的科学记数法，指数用 E 表示
%g	紧凑的十进制记数法或科学记数法，指数用 e 表示
%G	紧凑的十进制记数法或科学记数法，指数用 E 表示
%xz	右对齐的某种格式 z，字段宽度为 x
%-xz	左对齐的某种格式 z，字段宽度为 x
%.yz	有 y 位小数的某种格式 z
%x.yz	有 y 位小数且字段宽度为 x 的某种格式 z
%%	百分号 % 本身

指定文字和数字的输出格式：printf 风格

多行输出：三引号字符串可以实现多行输出 ball_print.py

```
v0 = 5
g = 9.8
t = 0.6
y = v0*t - 1/2 * g*t**2
print("""At t = %f s, a ball with
initial velocity v0 = %.3E m/s
is located at the height %.2f m.""" % (t, v0, y))
```

```
>>>
At t = 0.600000 s, a ball with
initial velocity v0 = 5.000E+00 m/s
is located at the height 1.24 m.
```

程序由语句构成

一般情况下每行程序一条语句

```
v0 = 5                          # 语句 1，赋值语句
g = 9.8                         # 语句 2
t = 0.6                         # 语句 3
y = v0*t - 1/2 * g*t**2         # 语句 4
print(y)                        # 语句 5
```

```
>>>
1.236
```

也可以把多条语句写在一行，中间需要用分号隔开

```
v0 = 5; g = 9.8; t = 0.6; y = v0*t - 1/2 * g*t**2;
print(y)
```

```
>>>
1.236
```

语法很重要，要求很苛刻

程序语句是给计算机的指令，不能包含错误

```
>>> v0 = 5
>>> prinnt(v0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'prinnt' is not defined
>>> print(V0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'V0' is not defined
>>> print(vo)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'vo' is not defined
>>> print(v0)
5
```

用好空格

下面的代码等价（哪种好看些？）

```
>>> v0=5
>>> v0   =   5
```

```
>>> v0=   5
>>> v0 = 5
```

程序的有些空格很重要

```
>>> counter = 1
>>> while counter <= 4:
...     counter = counter + 1 # 正确的语法：四个空格
...
>>> while counter >= 0:
... counter = counter - 1     # 错误的语法
  File "<stdin>", line 2
    counter = counter - 1     # 错误的语法
    ^^^^^^^
IndentationError: expected an indented block after 'while' sta
```

程序是对数据的处理，包含输入、处理和输出

```
v0 = 3; t = 0.6; g = 9.8
y = v0*t - 1/2 * g*t**2
v = v0 - g*t
print('y =', y)
print('v =', v)
```

```
>>>
y = 0.03599999999999981
v = -2.88
```

- 输入：也可使用 input 从控制台读取用户输入的内容

  ```
  v0 = 3; t = 0.6; g = 9.8
  ```

- 处理

  ```
  y = v0*t - 1/2 * g*t**2
  v = v0 - g*t
  ```

- 输出

  ```
  print('y =', y)
  print('v =', v)
  ```

提要

Python 中一切都是对象

变量指向对象

```
>>> a = 5                          # a 指向一个整型对象
>>> type(a)
<class 'int'>
>>> b = 9                          # b 指向一个整型对象
>>> c = 9.0                        # c 指向一个浮点数对象
>>> type(c)
<class 'float'>
>>> d = b/a                        # d 指向 int/int => 浮点数对象
>>> type(d)
<class 'float'>
>>> e = c/a                        # e 指向一个浮点数对象
>>> s = 'b/a = %g' % (b/a) # s 指向一个字符串对象
>>> type(s)
<class 'str'>
```

Python 中一切都是对象

可以转换对象类型

```
>>> a = 3          #a是 int
>>> print(a)
3
>>> b = float(a)#b是 float
>>> print(b)
3.0
>>> c = 3.934     #c是 float
>>> print(c)
3.934
>>> d = int(c)    #d是 int
>>> print(d)
3
```

```
>>> d = round(c)#d是 int
>>> print(d)
4
>>> e = str(c)   #e是 str
>>> type(e)
<class 'str'>
>>> f = '-4.2'   #f是 str
>>> type(f)
<class 'str'>
>>> g = float(e)#g是 float
>>> type(g)
<class 'float'>
```

提要

算数表达式

计算 $\frac{5}{9} + 2 \times 3^2/2$

```
>>> # 算数表达式和数学类似:
>>> 5/9 + 2*3**2/2
9.555555555555555

>>> # 计算过程遵循运算符的优先级和结合性:
>>> t1 = 5/9
>>> t2 = 3**2
>>> t3 = 2*t2
>>> t4 = t3/2
>>> t5 = t1 + t4

>>> # 也可以使用 () 改变计算顺序或者增加可读性:
>>> 5/9 + (2*(a**4))/2
81.55555555555556
```

使用标准数学函数：导入 math 库

导入 math 库后可以使用标准数学函数，如 sin，cos，ln 等

```
>>> import math
>>> r = math.sqrt(2)
>>> r
1.4142135623730951

>>> from math import sqrt
>>> r = sqrt(2)
>>> r
1.4142135623730951

>>> from math import *
>>> r = sqrt(2)
>>> r
1.4142135623730951
```

计算机的表示和计算误差

```
>>> v1 = 1/49 * 49
>>> v2 = 1/51 * 51
>>> print('%.16f %.16f' % (v1, v2))
0.9999999999999999 1.0000000000000000
```

- 大多数实数在计算机中都是非精确表示，误差在 10^{-16} 数量级
- 这种小误差可能会导致非正确的结果，也可能不会
- 注意：实数的计算机表示和计算都是近似的

测试

```
>>> a = 1
>>> b = 2
>>> c = 3
>>> a + b == c
True
```

```
>>> a = 0.1
>>> b = 0.2
>>> c = 0.3
>>> a + b == c
False
```

计算机的表示和计算误差

使用近似值判断浮点数相等

```
>>> a = 0.1; b = 0.2 ; c = 0.3
>>> abs(a + b - c) < 1E-15
True
```

误差的另一个例子：计算双曲正弦函数 $\sinh = \frac{1}{2}\left(\mathrm{e}^{x} - \mathrm{e}^{-x}\right)$

```
>>> from math import sinh, exp, e, pi
>>> x = 2*pi
>>> r1 = sinh(x)
>>> r2 = 0.5*(exp(x) - exp(-x))
>>> r3 = 0.5*(e**x - e**(-x))
>>> print('%.16f\n%.16f\n%.16f' % (r1,r2,r3))
267.7448940410164369
267.7448940410164369
267.7448940410163232
```

课堂练习

1 相对论质量　　文件名：relativity.py

根据相对论，一个物体的质量取决于其运动速度：

$$m = \frac{m_0}{\sqrt{1 - \left(\frac{v}{c}\right)^2}}$$

其中 m_0 和 m 分别是物体静止和运动时的质量，v 是物体运动速度，$c = 299792458$ m/s 是真空中的光速。编写程序计算某静止时的质量为 1 kg 的物体，以 0.6 倍光速运动时的质量，具体要求如下：

- 同时使用多行和单行注释
- 指定文字和数字的输出格式
- 分别使用标准库中的 sqrt 函数和 0.5 次方实现

提要

Python 支持复数计算

```
>>> x = complex(2, -3)
>>> x
(2-3j)
>>> y = 4 + 5j
>>> y
(4+5j)
>>> x*y
(23-2j)
>>> x/y
(-0.17073170731707318-0.5365853658536587j)
>>> x.real
2.0
>>> x.imag
-3.0
```

提要

Python 支持符号计算

竖直上抛

```
>>> from sympy import *
>>> t, v0, g = symbols('t v0 g')
>>> y = v0*t - Rational(1,2)*g*t**2
>>> dydt = diff(y, t); dydt                    # 一阶导数
-g*t + v0
>>> print('acceleration:', diff(y, t, t))     # 二阶导数
acceleration: -g
>>> y2 = integrate(dydt, t)                    # 积分
>>> y2
-g*t**2/2 + t*v0
>>> roots = solve(y, t)              # 求解 y = 0 时 t 的根
>>> roots
[0, 2*v0/g]
>>> y.subs(t, roots[1])              # 代回验证
0
```

Python 支持符号计算

三角函数

```
>>> from sympy import *
>>> x, y = symbols('x y')
>>> f = -sin(x)*sin(y) + cos(x)*cos(y)
>>> f
-sin(x)*sin(y) + cos(x)*cos(y)
>>> simplify(f)
cos(x + y)
>>> g = sin(x)**2 + cos(x)**2
>>> g
sin(x)**2 + cos(x)**2
>>> simplify(g)
1
>>> expand(sin(x+y), trig=True) # trig 指明是三角函数
sin(x)*cos(y) + sin(y)*cos(x)
```

课堂练习

2 竖直上抛的符号推导和计算　　文件名：ball_sym.py

根据竖直上抛的运动公式

$$y = v_0 t - \frac{1}{2} g t^2$$

编写程序，应用符号计算，完成以下任务：

- 求小球到达最高点的时间，即导数为零对应的时间。
- 求小球到达某位置 y 的时间，即 $v_0 t - \frac{1}{2} g t^2 - y = 0$ 对应的时间。
- 将 $v_0 = 5$ m/s，$g = 9.8$ m/s^2，$y = 1$ m 代入表达式求相应的 t。

The End!