

## 实验 1 Python 编程环境的使用

### 1. 实验目的

- (1) 了解 Python 编程环境，进行程序设计的基本训练
- (2) 熟悉 Python 语言的使用方式，编写简单 python 程序，包括编写和运行基本的输入、输出和数值计算程序。
- (3) 会定义和调用函数

### 2. 实验任务

#### 实验任务 1-1 圆台体积计算

编写圆台体积计算程序，即通过输入上底半径  $r$ 、下底半径  $R$ 、高  $h$  计算圆台体积。将计算结果保留两位小数。提示：圆台体积公式为：

$$V = \frac{1}{3}\pi h(R^2 + r^2 + R \cdot r)$$

输入格式：

输入数据包含三行，分别依次为上底半径  $r$ 、下底半径  $R$ 、和高  $h$ ，可能包含小数。

输出格式：

输出数据包含一行，为一个数  $V$ ，表示圆台体积。保留两位小数。

实验目的：

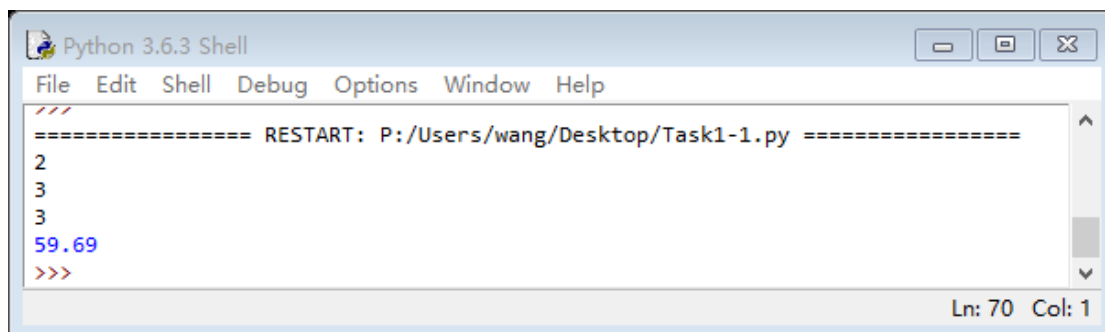
本实验帮助理解变量的赋值、程序的输入输出方法，以及简单的数值计算。

实验指导：

1. Python 中通过 `input()` 函数得到的输入为字符串变量，因此在计算前需要使用 `float()` 函数转化为浮点型，如 `r = float(r)`。
2. Python 中乘方运算可以使用 `**` 符号实现，如  $r^2$  可以用 `r ** 2` 计算。
3. 在输出浮点型数据时，我们往往需要保留确定的小数位数。为了实现这一目的，我们可以在 `print()` 时在数据前加上 `'%.nf' %`，其中  $n$  为需要保留的小数位数。例如：`print('%.3f' % a)`。我们也可以使用 `format()` 函数，如 `print("{:.3f}".format(a))` 也可以达成同样效果。

有关这一方法的详细解释，可参见 Python 官方文档 [printf-style String Formatting](#) 及 [Format String Syntax](#) 的有关说明。

参考运行结果：



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:/Users/wang/Desktop/Task1-1.py =====
2
3
3
59.69
>>>
```

## 实验任务 1-2 小明的实验数据

小明同学最近正在做基础物理实验，每次都对数据处理的大量计算十分头疼，所以想借助 python 来帮助自己进行数据处理，今天小明在测钢丝直径时测量了 3 次直径。请读入这 3 次直径值，要求输出平均值和标准差。

注：设共有  $k$  个数据，令  $x_i$  表示第  $i$  个数据， $\bar{x}$  表示数据平均值， $s(\bar{x})$  表示数据的标准差，则平均值和标准差的计算方法为：

$$\bar{x} = \frac{\sum_{i=1}^k x_i}{k}, s(\bar{x}) = \sqrt{\frac{\sum_{i=1}^k (x_i - \bar{x})^2}{k - 1}}$$

输入格式：

输入数据共一行，表示 3 次测量的直径值，中间用空格隔开。

输出格式：

输出数据共一行，表示平均值和标准差，中间用空格隔开，保留 2 位小数。

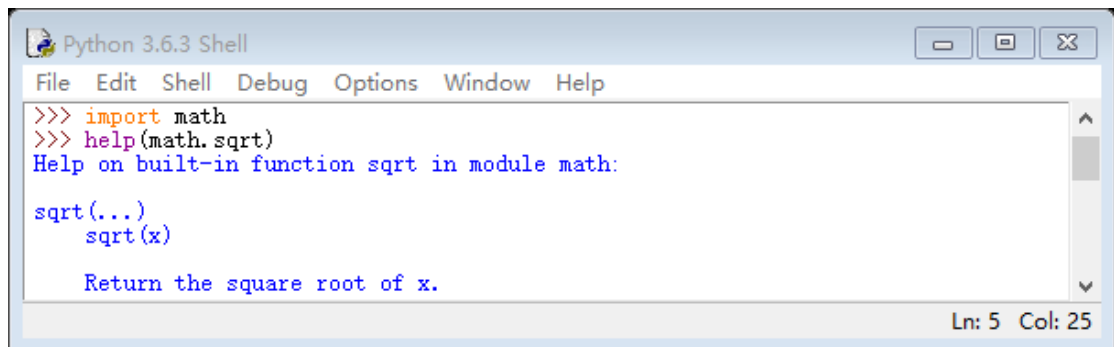
实验目的：

本实验旨在帮助理解库函数的调用。

实验指导：

1. Python 中通过 `input()` 函数获得的是字符串变量，若一行输入数据有多个。可以用 `a,b,... = input().split()` 按照空格将输入的多个值分开，分别赋给前面的变量。
2. 在 Python 中使用开方等数学运算需要导入 `math` 包，可以写 `from math import sqrt`，此后就可以直接使用 `sqrt()` 函数，如 `s = sqrt(a)`；也可以写 `import math`，这样就可以将 `math` 包中的全部函数引入，但是使用函数时要使用成 `s = math.sqrt(a)`。

3. 我们在 Python 后序学习中会经常使用别人已经编写好的函数。想要具体了解这些函数的使用方法可以去[官网](#)查询，或者在 IDLE 中使用 `help()` 函数，以本题为例：



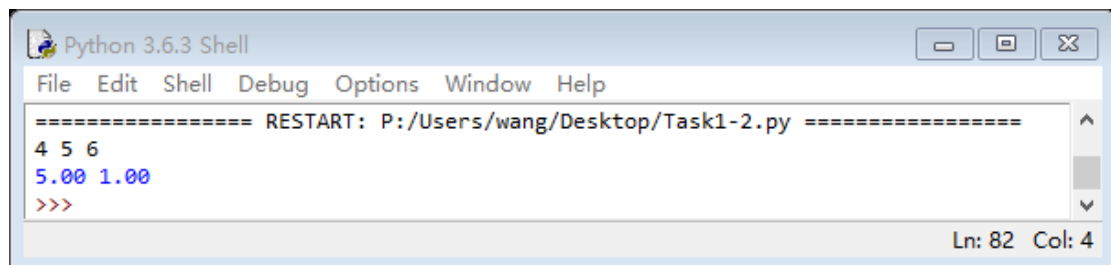
```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
>>> import math
>>> help(math.sqrt)
Help on built-in function sqrt in module math:

sqrt(...)
    sqrt(x)

    Return the square root of x.
```

Ln: 5 Col: 25

参考运行结果：



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:/Users/wang/Desktop/Task1-2.py =====
4 5 6
5.00 1.00
>>>
```

Ln: 82 Col: 4

### 实验任务 1-3 简单密码

密码学 (Cryptography) 是和计算机发展紧密联系的一门学科。在很长时间的的发展过程中, 密码由简单变得复杂, 但是密码离不开加密和解密两个过程。假设存在以下加密方式, 将一组已知信息 (一个十进制数) 通过乘以  $a$ , 然后减去  $b$ , 再乘以  $c$ , 最后加上  $d$  的方式进行加密。现请你编写一个函数, 读取用户输入, 调用该函数并输出实现该过程。

输入格式:

输入数据共五行, 依次为原码,  $a$ ,  $b$ ,  $c$ ,  $d$ 。

输出格式:

输出共一行, 为 The cyphertext is  $m$ ., 其中  $m$  以密文代替。

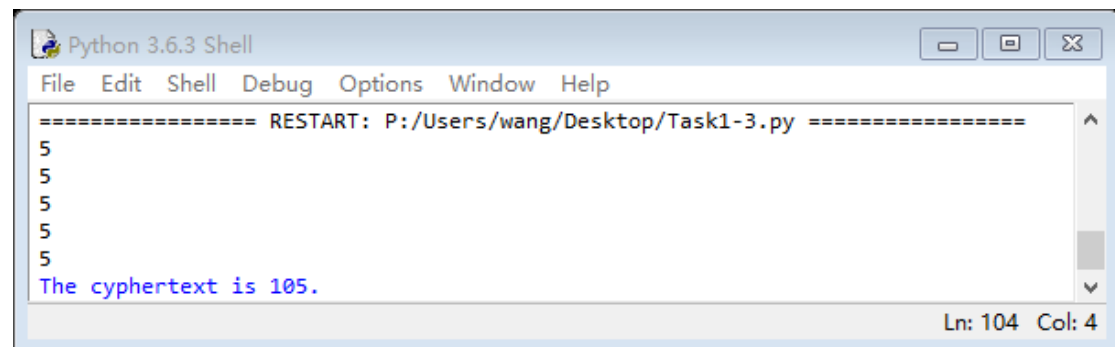
实验目的:

理解数值 (任意类型) 的运算, 学习函数的定义方法。

实验指导:

Python 中不可将  $+$  在字符串与数字之间使用, 也就是说使用 `'a'+3` 既不能得到 `'a3'` 也不能得到 `'d'`。如希望得到前者, 数字 `3` 须转化成字符串型, 即使用 `'a'+str(3)`。如希望得到后者, 请参考下一题。

参考运行结果:



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:/Users/wang/Desktop/Task1-3.py =====
5
5
5
5
5
The cyphertext is 105.
Ln: 104 Col: 4
```

## 实验任务 1-4 ASCII 码

在计算机中,用若干位二进制符号表示数字、字母、命令以及特殊符号的方法称为字符编码,又称作 ASCII 码(American Standard Code for Information Interchange, 美国国家信息交换标准码)。常用字符有 128 个,其十进制 ASCII 码值从 0 到 127(如下表所示),其中 0 为空字符,1~31 表示的是各种控制字符。ASCII 码用 7 位二进制符号来表示字符和命令,编码为  $000\ 0000_{\text{bin}} \sim 111\ 1111_{\text{bin}}$ ,为书写方便,一般缩写成两位十六进制编码,为  $00_{\text{hex}} \sim 7F_{\text{hex}}$ 。例如:“a”字符的 ASCII 码编码为  $61_{\text{hex}}$ ,其十进制 ASCII 码值为 97,“A”字符的 ASCII 码编码为  $41_{\text{hex}}$ ,其十进制 ASCII 码值为 65。

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	\$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: [www.LookupTables.com](http://www.LookupTables.com)

请你试用 Python 编写一个函数,该函数完成把大写字母转化为对应的小写字母的功能。获得从键盘输入的一个大写字母,调用你所编写的函数,能将该字符串转化为相应的小写字母进行输出。

输入格式:

输入为一个字符,表示待转化的大写字母。

输出格式:

输出为一个字符,表示转化后的小写字母。

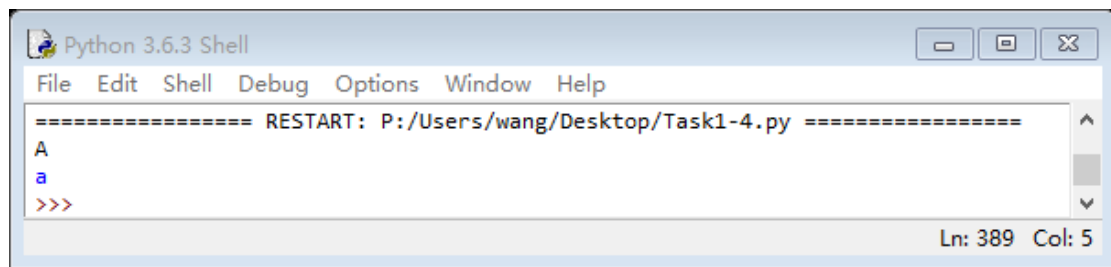
### 实验目的：

理解函数的定义方法。

### 实验指导：

Python 提供了 `ord()` 函数和 `chr()` 函数实现字符和 ASCII 码的转换：`ord()` 函数可用于获取含单个字符的字符串中该字符的 ASCII 码值，如 `ord('c')` 的结果为 99，`ord('cc')` 则会报错；`chr()` 函数则相反，可获得一个 ASCII 码值对应的字符，如 `chr(99)` 的结果为 'c'。

### 参考运行结果：



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:/Users/wang/Desktop/Task1-4.py =====
A
a
>>>
Ln: 389 Col: 5
```

## 实验任务 1-5 打点计时器（选做）

打点计时器是一种测量短暂时间的工具。如果运动物体带动的纸带通过打点计时器，在纸带上打下的点就记录了物体运动的时间，纸带上的点也相应的表示出了运动物体在不同时刻的位置。

请你编写一个程序，通过分析自由落体运动上方连接的纸带，输入打点计时器以 0.05s 的时间间隔打出六个计数点的位置信息，计算在中间四个点的速度，利用用速度信息使用线性回归计算加速度，并与标准值相比较（ $9.8\text{m/s}^2$ ），计算相对误差。

输入格式：

输入数据为六行，依次给出实验后纸带上连续六个计数点（不一定是最初六个）距起始点的距离（单位：厘米）。

输出格式：

对于每组数据，输出两行：

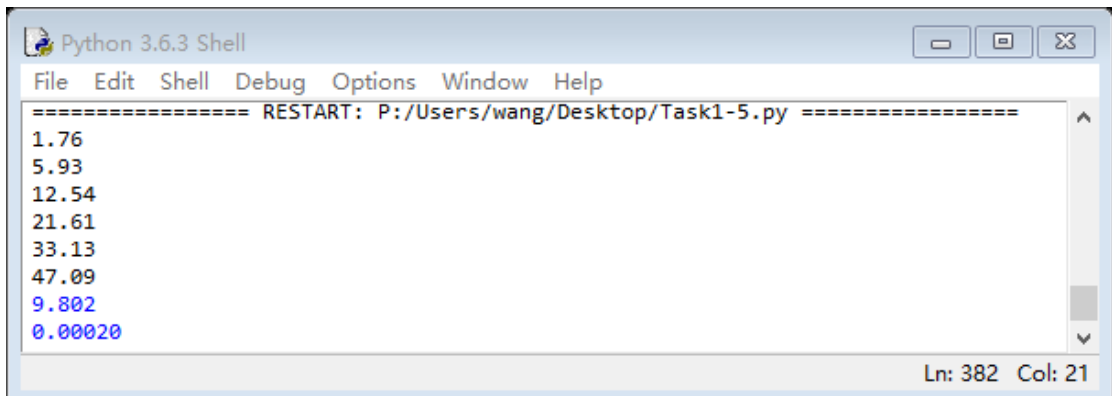
第一行为纸带的加速度，单位  $\text{m/s}^2$ ，保留三位小数。

第二行为速度与标准值  $9.8\text{m/s}^2$  的相对误差的绝对值，保留五位小数。

实验指导：

1. 输入的数据单位是厘米，输出单位是  $\text{m/s}^2$ ，注意单位换算
2. 一元线性回归时，斜率公式为  $b = \frac{\bar{y}\bar{x} - \bar{x}\bar{y}}{\bar{x}^2 - \bar{x}^2}$ 。

参考运行结果：



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:/Users/wang/Desktop/Task1-5.py =====
1.76
5.93
12.54
21.61
33.13
47.09
9.802
0.00020
Ln: 382 Col: 21
```