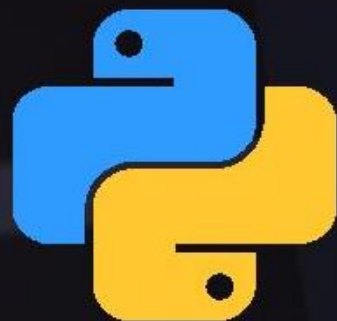


Python程序语言入门与应用



python

Life is short, use Python
人生苦短，我用Python




课程回顾



Python程序语法元素分析

-  温度转换实例

-  缩进，注释，命名与保留字，字符串，赋值，分支语句与循环语句，函数

-  turtle库绘制蟒蛇

-  Python标准库的导入与使用 (import)

-  模块化编程和面向对象



新乡医学院

Python程序语言入门与应用

深入Python

第三章 Python基本数据类型



郭长江

changjiangguo@xxmu.edu.cn

生命科学技术学院

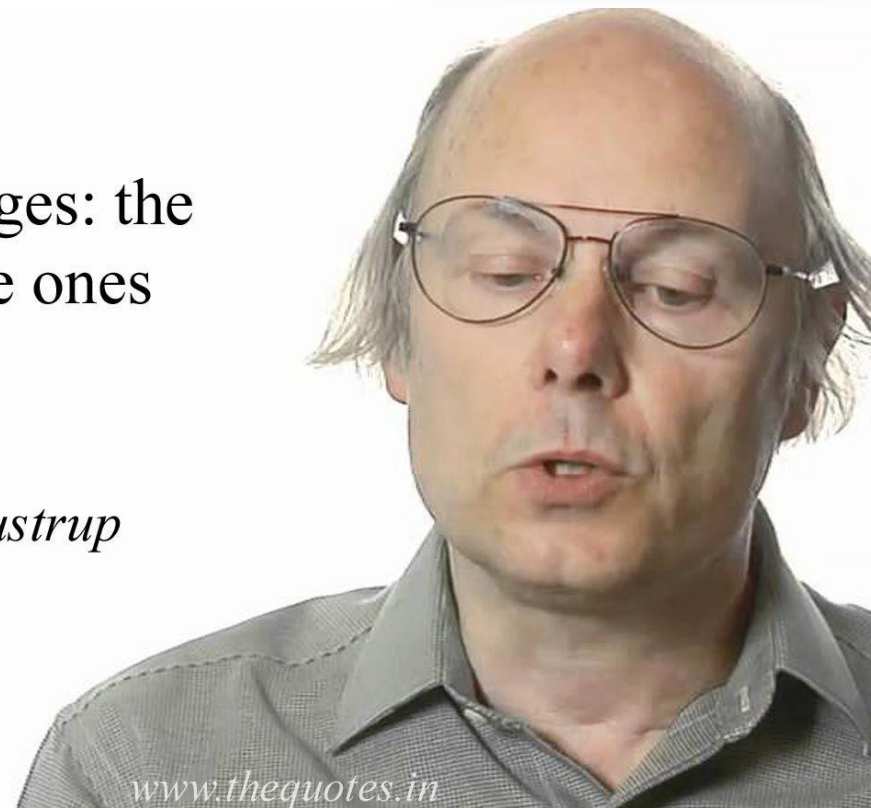
新乡医学院





There are only two kinds of languages: the ones people complain about and the ones nobody uses.

Bjarne Stroustrup



只有两种编程语言：一种是经常被骂的，一种是没人使用的。

——本贾尼·斯特劳斯特卢普（C++语言之父）



学习目标



基本要求

掌握


-  Python语言基本数据类型的概念和使用

理解

-  字符串类型的格式化操作方法和应用

了解

-  3种数字类型在计算机中的表示方法

-  math库的使用



本课概要



第3章 Python基本数据类型

- 3.1 数字类型
- 3.2 数字类型的操作
- 3.3 math库的使用
- 3.4 实例：天天向上的力量
- 3.5 字符串类型及其操作
- 3.6 字符串类型的格式化
- 3.7 实例：文本进度条



3.1 数字类型



3.1 数字类型



❧ 数字类型

❧ 表示数字或数值的数据类型

❧ Python中有3种数字类型

❧ 整数

❧ 浮点数

❧ 复数



3.1 数字类型



❖ 整数类型 (int)

❖ 与数学中整数的概念一致

❖ 如1010, 99, -217, 0b101, 0o711, 0x9a

❖ 共有4种进制表示

❖ 二进制、八进制、**十进制**（默认）、十六进制

| 进制种类 | 引导符号 | 描述 |
|------------|----------|--------------------------------|
| 二进制 | 0b或0B | 由字符0和1组成, 如0b10101 |
| 八进制 | 0o或0O | 由字符0到7组成, 如0o765 |
| 十进制 | 无 | 由字符0到9组成, 如9876, -54321 |
| 十六进制 | 0x或0X | 由字符0到9、a到f、A到F组成, 如0xABC |



3.1 数字类型



❖ 整数类型 (int)

- ❖ 可正可负，没有取值范围限制
- ❖ 理论范围 $[-\infty, \infty]$ ，实际受限于计算机内存
- ❖ 内置函数`pow(x, y)`计算 x^y

```
>>> pow(2,100)
```

```
1267650600228229401496703205376
```

```
>>> pow(2,500)
```

```
1415461031044954789001553.....
```



3.1 数字类型



❖ 浮点数类型 (float)

- ❖ 与数学中实数的概念一致，表示带有小数的数值
- ❖ 浮点数取值范围和小数精度都存在限制，但常规计算可忽略（通过[sys.float_info](#)查看）

❖ 表示方法

❖ 十进制表示，如0.0, -77.1, 3.1415,

❖ 科学计数法，如96e4, 4.3e-3, 9.6E5 $\langle a \rangle e \langle b \rangle = a * 10^b$

```
>>> import sys
>>> sys.float_info
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308,
min=2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15,
mant_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)
```

取值范围数量级约 -10^{308} 至 10^{308} ，精度数量级 10^{-16}



3.1 数字类型



❖ 浮点数类型 (float)

$$\langle a \rangle e \langle b \rangle = a * 10^b$$

❖ 浮点数科学计数法表中的系数最长为16个数字

❖ 浮点数运算结果最长可输出17个数字

>>> 3.1415926535897924

3.1415926535897922

>>> 987654321123456.789

987654321123456.8

❖ 计算机仅能够提供15个数字的准确性

❖ 浮点数无法进行极高精度的数学运算

>>> 3.141592653*12.34567898

38.78509437986454

>>> 3141592653*1234567898

3878509437986453394



3.1 数字类型



❧ 浮点数类型 (float)

❧ 浮点数间运算存在**不确定尾数**，不是bug

```
>>> 0.1 + 0.3
```

```
0.4
```

```
>>> 0.1 + 0.2
```

```
0.30000000000000004
```

❧ 产生原因

❧ 二进制表示小数，可以无限接近，但不完全相同

```
>>> 0.1 + 0.2 == 0.3
```

```
False
```

```
>>> round(0.1+0.2, 1) == 0.3
```

```
True
```

❧ **round**(x, d)

❧ 对x四舍五入，d是小数截取位数

❧ 浮点数间运算及比较用round()函数辅助

❧ 不确定尾数一般发生在 10^{-16} 左右，round()十分有效



3.1 数字类型



❖ 复数类型 (complex)

❖ 与数学中复数的概念一致

❖ $a+bj$ 被称为复数

❖ a 是实部, b 是虚部

❖ $j^2 = -1$

❖ 如 $z = 1.23e-4 + 5.6e+89j$

❖ `z.real` 获得实部

❖ `z.imag` 获得虚部

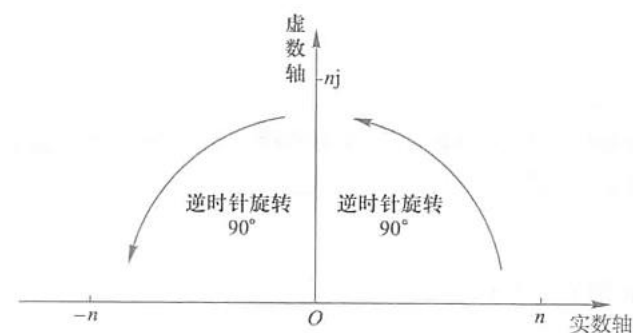


图 3.1 虚数的表示



3.2 数字类型的操作



3.2 数字类型的操作



数字类型的操作方法

 数值运算操作符

 数值运算函数

 类型转换函数



3.2 数字类型的操作



数值运算操作符

操作符是完成运算的一种符号体系

9种内置操作符

| 操作符 | 描述 |
|----------|--|
| $x + y$ | x与y之和, 如 $1 + 2 = 3$ |
| $x - y$ | x与y之差, 如 $3 - 2 = 1$ |
| $x * y$ | x与y之积, 如 $2 * 3 = 6$ |
| x / y | x与y之商, 如 $10 / 4 = 2.5$ |
| $x // y$ | x与y之整数商, 即不大于其商的最大整数, 如 $10 // 4 = 2$ |
| $x \% y$ | x与y之商的余数, 也称为 模运算 , 如 $10 \% 4 = 2$ |
| $-x$ | x的负值, 即 $x * (-1)$, 如 $-(-1) = 1$ |
| $+x$ | x本身 |
| $x ** y$ | x的y次幂, 即 x^y , 如 $2 ** 3 = 8$ 、 $4 ** 0.5 = 2$ |



3.2 数字类型的操作



数值运算操作符

操作符运算的结果可能改变数字类型

整数 \rightarrow 浮点数 \rightarrow 复数

是否改变取决于其数学意义上的运算结果

```
>>> 100/3
33.333333333333336
>>> 100/2
50.0
>>> 100//3
33
>>> 123 + 4
127
>>> 123 + 4.0
127.0
```

```
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
>>> type(1 + 1j)
<class 'complex'>
```

type()函数查看数字类型



3.2 数字类型的操作



❖ 二元数值运算操作符

❖ 数值运算操作符对应有增强的赋值操作符

❖ $x \text{ op} = y$ 相当于 $x = x \text{ op } y$

❖ 简化代码表达

| 操作符 | 描述 |
|--------------------|---|
| $x \text{ += } y$ | $x = x + y$, 如 $x = 1, x += 1 \rightarrow x = 2$ |
| $x \text{ -= } y$ | $x = x - y$, 如 $x = 2, x -= 1 \rightarrow x = 1$ |
| $x \text{ *= } y$ | $x = x * y$, 如 $x = 1, x *= 2 \rightarrow x = 2$ |
| $x \text{ /= } y$ | $x = x / y$, 如 $x = 2, x /= 2 \rightarrow x = 1$ |
| $x \text{ //= } y$ | $x = x // y$, 如 $x = 10, x //= 3 \rightarrow x = 3$ |
| $x \text{ %= } y$ | $x = x \% y$, 如 $x = 10, x \% = 3 \rightarrow x = 1$ |
| $x \text{ **=} y$ | $x = x ** y$, 如 $x = 2, x ** = 3 \rightarrow x = 8$ |



3.2 数字类型的操作



数值运算操作符

- 数值运算操作符存在优先级
- 不清楚优先级时多使用括号

| 操 作 符 | 描 述 |
|---------------------------------|--------------|
| ** | 指数（最高优先级） |
| * / % // | 乘，除，取模和取整除 |
| + - | 加法减法 |
| = %= /= //= -= += *= **= | 赋值运算符（最低优先级） |

```
>>> 30-3**2+8//3**2*10  
21
```

```
>>> (30-(3**2))+((8//(3**2))*10)  
21
```



3.2 数字类型的操作



数值运算函数

6个内置数值运算函数

| 函 数 | 描 述 |
|---|--|
| abs (x) | x的绝对值, abs (-10.01) 结果为 10.01 |
| divmod (x, y) | 商余, (x//y, x%y), 同时输出商和余数 divmod (10, 3) 结果为 (3, 1) |
| pow (x, y[, z]) | 幂余, (x**y)%z, [...]表示参数z可省略 pow (3, pow (3, 99), 10000) 结果为 4587 |
| round (x[, ndigits) | 四舍五入, d是保留小数位数, 默认值为0 round (-10.123, 2) 结果为 -10.12 |
| max (x ₁ , x ₂ , ..., x _n) | 最大值, 返回x ₁ , x ₂ , ..., x _n 中的最大值, n不限 max (1, 9, 5, 4 3) 结果为 9 |
| min (x ₁ , x ₂ , ..., x _n) | 最小值, 返回x ₁ , x ₂ , ..., x _n 中的最小值, n不限 min (1, 9, 5, 4 3) 结果为 1 |



3.2 数字类型的操作



☼ 数字类型转换函数

☼ 3个内置类型转换函数

| 函 数 | 描 述 |
|---------------------------|---|
| int (x) | 将x变成整数，舍弃小数部分，x可以是浮点数和字符串 int(123.45) 结果为123； int("123") 结果为123 |
| float (x) | 将x变成浮点数，增加小数部分，x可以是整数和字符串 float(12) 结果为12.0； float("1.23") 结果为1.23 |
| complex (re[, im]) | 生成一个复数，实部为re，虚部为im，re可以是整数、浮点数和字符串 complex(4) 结果为 4 + 0j |

```
>>> int("1.0")
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    int("1.0")
ValueError: invalid literal for int() with
base 10: '1.0'
```

```
>>> float(10 + 99j)
Traceback (most recent call last):
  File "<pyshell#35>", line 1, in <module>
    float(10 + 99j)
TypeError: can't convert complex to float
>>> float((10 + 99j).imag)
99.0
```



3.3 math库的使用



3.3 math库的使用



❧ 函数库

- ❧ Python语言最重要的特点：利用函数库编程
- ❧ 函数库Library，也称为模块Module
- ❧ 默认支持的函数库称为标准函数库或内置函数库
- ❧ 第三方提供需要进行安装的函数库称为第三方函数库
- ❧ 优点：不用“重复造轮子” & 模块编程

❧ 数学计算标准函数库math

- ❧ 支持整数和浮点数运算
- ❧ 4个数学常数 + 44个函数



3.3 math库的使用



导入方式

```
import math
```

```
import math as m
```

```
from math import floor, ceil
```

```
from math import *
```

函数调用方式

```
math.floor()
```

```
m.floor()
```

```
floor()
```

```
# 查看所有函数名列表
```

```
dir(math)
```

```
# 查看所有定义及函数原型
```

```
help(math)
```

```
# 查看函数说明
```

```
help(math.floor)
```



3.3 math库的使用



数学常数 (4个)

| 常 数 | 数学表示 | 描 述 |
|-----------------------|----------|----------------------------|
| <code>math.pi</code> | π | 圆周率, 值为3.141592653589793 |
| <code>math.e</code> | e | 自然对数, 值为2.718281828459045 |
| <code>math.inf</code> | ∞ | 正无穷大 |
| <code>math.nan</code> | | 非浮点数标记, NaN (Not a Number) |



3.3 math库的使用



函数 (44个)

 数值表示函数 (16个)

 幂对数函数 (8个)

 三角运算函数 (16个)

 高等特殊函数 (4个)



3.3 math库的使用



数值表示函数 (16个)

| 函数 | 数学表示 | 描述 |
|------------------------------------|---------------------|--|
| <code>math.fabs(x)</code> | $ x $ | 返回x的绝对值 |
| <code>math.fmod(x)</code> | $x \% y$ | 返回x与y的模 |
| <code>math.fsum([x, y,...])</code> | $x + y + \dots$ | 浮点数精确求和 |
| <code>math.ceil(x)</code> | $\lceil x \rceil$ | 向上取整, 返回不小于x的最小整数 |
| <code>math.floor(x)</code> | $\lfloor x \rfloor$ | 向下取整, 返回不大于x的最小整数 |
| <code>math.factorial(x)</code> | $x!$ | 返回x的阶乘, 如x非整数, 返回ValueError |
| <code>math.gcd(x, y)</code> | | 返回x与y的最大公约数 |
| <code>math.frexp(x)</code> | $x = m * 2^e$ | 返回(m,e), 当x=0, 返回(0.0, 0) |
| <code>math.ldexp(x)</code> | $x * 2^i$ | 返回x*2 ⁱ 的运算值, math.frexp(x)的反运算 |
| <code>math.modf(x)</code> | | 返回x的整数部分和小数部分 |
| <code>math.trunc(x)</code> | | 返回x的整数部分 |
| <code>math.copysign(x, y)</code> | $ x * y / y$ | 用数值y的正负号替换数值x的正负号 |
| <code>math.isclose(x, y)</code> | | 比较x与y的相似性, 返回True或False |
| <code>math.isfinite(x)</code> | | 当x不是无穷大或NaN时返回True, 否则返回False |
| <code>math.isinf(x)</code> | | 当x为正负无穷大时返回True, 否则返回False |
| <code>math.isnan(x)</code> | | 当x是NaN时返回True, 否则返回False |



3.3 math库的使用



数值表示函数 (示例)

```
>>> 0.1 + 0.2 + 0.3
0.6000000000000001
>>> math.fsum([0.1, 0.2, 0.3])
0.6
>>> math.fmod(10, 3)
1.0
>>> math.ceil(1.5)
2
>>> math.floor(1.5)
1
>>> math.factorial(10)
3628800
>>> math.gcd(15, 20)
5
>>> math.modf(1.5)
(0.5, 1.0)
```



3.3 math库的使用



幂对数函数（8个）

| 函 数 | 数学表示 | 描 述 |
|----------------------------------|------------------------|---------------------------------|
| <code>math.pow(x, y)</code> | x^y | 返回x的y次幂 |
| <code>math.exp(x)</code> | e^x | 返回e的x次幂 |
| <code>math.expm1(x)</code> | $e^x - 1$ | 返回e的x次幂减1 |
| <code>math.sqrt(x)</code> | \sqrt{x} | 返回x的平方根 |
| <code>math.log(x[, base])</code> | $\log_{\text{base}} x$ | 返回以base为底x的对数值，无base时返回 $\ln x$ |
| <code>math.log1p(x)</code> | $\ln(x+1)$ | 返回x+1的自然对数值 |
| <code>math.log2(x)</code> | $\log_2 x$ | 返回以2为底x的对数值 |
| <code>math.log10(x)</code> | $\log_{10} x$ | 返回以10为底x的对数值 |



3.3 math库的使用



三角运算函数 (16个)

| 函 数 | 数 学 表 示 | 描 述 |
|-----------------|----------------------------|----------------------------|
| math.degree(x) | | 角度 x 的弧度值转角度值 |
| math.radians(x) | | 角度 x 的角度值转弧度值 |
| math.hypot(x,y) | $\sqrt{x^2 + y^2}$ | 返回(x,y)坐标到原点(0,0)的距离 |
| math.sin(x) | $\sin x$ | 返回 x 的正弦函数值, x 是弧度值 |
| math.cos(x) | $\cos x$ | 返回 x 的余弦函数值, x 是弧度值 |
| math.tan(x) | $\tan x$ | 返回 x 的正切函数值, x 是弧度值 |
| math.asin(x) | $\arcsin x$ | 返回 x 的反正弦函数值, x 是弧度值 |
| math.acos(x) | $\arccos x$ | 返回 x 的反余弦函数值, x 是弧度值 |
| math.atan(x) | $\arctan x$ | 返回 x 的反正切函数值, x 是弧度值 |
| math.atan2(y,x) | $\arctan y/x$ | 返回 y/x 的反正切函数值, x 是弧度值 |
| math.sinh(x) | $\sinh x$ | 返回 x 的双曲正弦函数值 |
| math.cosh(x) | $\cosh x$ | 返回 x 的双曲余弦函数值 |
| math.tanh(x) | $\tanh x$ | 返回 x 的双曲正切函数值 |
| math.asinh(x) | $\operatorname{arcsinh} x$ | 返回 x 的反双曲正弦函数值 |
| math.acosh(x) | $\operatorname{arccosh} x$ | 返回 x 的反双曲余弦函数值 |
| math.atanh(x) | $\operatorname{arctanh} x$ | 返回 x 的反双曲正切函数值 |



3.3 math库的使用



高等特殊函数（4个）

| 函 数 | 数 学 表 示 | 描 述 |
|----------------|--|---------------------------------------|
| math.erf(x) | $\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ | 高斯误差函数，应用于概率论、统计学等领域 |
| math.erfc(x) | $\frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$ | 余补高斯误差函数，math.erfc(x)=1 - math.erf(x) |
| math.gamma(x) | $\int_0^\infty x^{t-1} e^{-x} dx$ | 伽玛（Gamma）函数，也叫欧拉第二积分函数 |
| math.lgamma(x) | ln(gamma(x)) | 伽玛函数的自然对数 |



好好学习
天天向上
毛泽东

3.4 实例：天天向上的力量



3.4 实例：天天向上的力量



☁ 问题分析 持续的价值

☁ 一年365天，每天进步一点点($x\%$)，
累计进步多少呢？

$$(1+x\%)^{365}$$

☁ 一年365天，每天退步一点点($x\%$)，
累计剩余多少呢？

$$(1-x\%)^{365}$$

$$1.01^{365}=37.8$$

$$0.99^{365}=0.03$$

如果等式一告诉我们，每天进步一点点，退步就少一点。

$$1.02^{365}=1377.4$$

$$0.98^{365}=0.0006$$

那么等式二则告诉我们，
只比你努力一点的人，其实，已经甩你太远。





3.4 实例：天天向上的力量



需求分析

- 数学公式或者计算器可以求解，似乎没必要用程序
- 如果是"三天打鱼两天晒网"呢？

$$1.01^{365} = 37.8$$

$$0.99^{365} = 0.03$$

如果等式一告诉我们，每天进步一点点，退步就少一点。

$$1.02^{365} = 1377.4$$

$$0.98^{365} = 0.0006$$

那么等式二则告诉我们，
只比你努力一点的人，其实，已经甩你太远。





3.4 实例：天天向上的力量



问题1： 1‰的力量

一年365天，每天进步1‰，累计进步多少呢？

$$1.001^{365}$$

一年365天，每天退步1‰，累计剩余多少呢？

$$0.999^{365}$$

```
# 3.1DayDayUp0.001.py
import math
dayup = math.pow((1.0 + 0.001), 365)
daydown = math.pow((1.0 - 0.001), 365)
print("向上: {:.2f}, 向下: {:.2f}.".format(dayup, daydown))
```

(运行结果) 向上: 1.44, 向下: 0.69.



3.4 实例：天天向上的力量



问题2： 5‰的力量

一年365天，每天进步5‰，累计进步多少呢？

$$1.005^{365}$$

一年365天，每天退步5‰，累计剩余多少呢？

$$0.995^{365}$$

```
# 3.1DayDayUp0.005.py
import math
dayup = math.pow((1.0 + 0.005), 365)
daydown = math.pow((1.0 - 0.005), 365)
print("向上: {:.2f}, 向下: {:.2f}.".format(dayup, daydown))
```

(运行结果) 向上: 6.17, 向下: 0.16.



3.4 实例：天天向上的力量



问题3： 1%的力量

一年365天，每天进步1%， 累计进步多少呢？

$$1.01^{365}$$

一年365天，每天退步1%， 累计剩余多少呢？

$$0.99^{365}$$

```
# 3.1DayDayUp0.01.py
import math
dayup = math.pow((1.0 + 0.01), 365)
daydown = math.pow((1.0 - 0.01), 365)
print("向上: {:.2f}, 向下: {:.2f}.".format(dayup, daydown))
```

(运行结果) 向上: 37.78, 向下: 0.03.



3.4 实例：天天向上的力量



问题4： 3天打鱼2天晒网

一年365天，一周5个工作日，每个工作日努力学习，提高1%，周末放任一下，退步1%，效果如何？

```
# 3.1DayDayUp5.2.py
import math
dayup, dayfactor = 1.0, 0.01
for day in range(365):
    if day % 7 in [0, 6]:
        dayup *= (1 - dayfactor)
    else:
        dayup *= (1 + dayfactor)
print("向上5天, 向下2天的力量: {:.2f}.".format(dayup))
```

(运行结果) 向上5天, 向下2天的力量: 4.63.



3.4 实例：天天向上的力量



问题5： 3天打鱼2天晒网

一年365天，每周末放任，退步1%，一周5个工作日，每个工作日提高多少，才能达到每天努力1%的效果？

```
# 3.1DayDayUp5.2up.py
import math
def dayUP(df):
    dayup = 1.0
    for day in range(365):
        if day % 7 in [0, 6]:
            dayup *= (1 - 0.01)
        else:
            dayup *= (1 + df)
    return dayup
dayup, dayfactor = 1.0, 0.01
while(dayUP(dayfactor) < 37.78):
    dayfactor += 0.001
print("每天努力的参数是：{:.3f}.".format(dayfactor))
```

(运行结果)
每天努力的参数是：0.019.



3.5 字符串类型及其操作



3.5 字符串类型及其操作



❖ 字符串类型

❖ 字符的序列表示，如abcedfg123456789

❖ 字符串表示

❖ 单引号：'xxx'（字符中可以有双引号）

❖ 双引号："xxx"（字符中可以有单引号）

❖ 三引号：'''xxx'''或者"""xxx"""（可以换行）

```
>>> print('单引号内可以使用双引号："')  
单引号内可以使用双引号："  
>>> print("双引号内可以使用单引号：'")  
双引号内可以使用单引号：'
```

```
>>> print('''  
三引号内可以换行  
可以使用单双引号：'""''')  
三引号内可以换行  
可以使用单双引号：'''
```



3.5 字符串类型及其操作



字符串类型

- input()函数将用户输入内容作为一个字符串类型
- 字符串可以索引字符： $[\pm N]$
- 字符串可以区间索引进行切片： $[\pm N:\pm M]$





3.5 字符串类型及其操作



❖ 特殊字符: \ (反斜线)

- ❖ 在字符串中表示转义, 与其后的字符组成新的含义
- ❖ \'表示单引号、\"表示双引号
- ❖ \\表示 \
- ❖ \n表示换行、\r表示回车键
- ❖ \t表示制表符、\b表示退格

| 转义字符 | 描述 |
|----------|---------------------------|
| \\(在行尾时) | 续行符 |
| \\ | 反斜杠符号 |
| '\' | 单引号 |
| '\"' | 双引号 |
| \a | 响铃 |
| \b | 退格(Backspace) |
| \e | 转义 |
| \000 | 空 |
| \n | 换行 |
| \v | 纵向制表符 |
| \t | 横向制表符 |
| \r | 回车 |
| \f | 换页 |
| \oyy | 八进制数yy代表的字符, 例如: \o12代表换行 |
| \xyy | 十进制数yy代表的字符, 例如: \x0a代表换行 |
| \other | 其它的字符以普通格式输出 |



3.5 字符串类型及其操作



字符串操作符

5种字符串基本操作符

| 操作符 | 描述 |
|-------------------|--|
| $x + y + \dots$ | 连接多个字符串x、y、..., 如 <code>'x'+'y'+'z'='xyz'</code> , <code>'1'+'2'='12'</code> |
| $x * n$ 或 $n * x$ | 复制n次字符串x, 如 <code>'x' * 5 = 'xxxxx'</code> |
| $x \text{ in } s$ | x是否是s的子串, 返回对应布尔值, 如 <code>'x' in 'xyz' = True</code> |
| $x[N]$ | x索引字符, 如 <code>'xyz'[2] = 'z'</code> |
| $x[N:M]$ | x索引切片, 如 <code>'xyz'[0:2] = 'xy'</code> |



3.5 字符串类型及其操作



字符串操作符

微实例：获取星期字符串

```
#3.2PrintWeekName.py
```

```
weekstr = "星期一星期二星期三星期四星期五星期六星期日"
```

```
weekid = eval(input("请输入星期数字 (1-7) : "))
```

```
pos = (weekid - 1) * 3
```

```
print(weekstr[pos:pos+3])
```

(运行结果)

请输入星期数字 (1-7) : 2

星期二

请输入星期数字 (1-7) : 5

星期五

请输入星期数字 (1-7) : 7

星期日



3.5 字符串类型及其操作



🔗 内置字符串处理函数

🔗 6种内置字符串处理函数

| 函 数 | 描 述 |
|----------------|--|
| len (x) | 返回字符串x的长度，如 <code>len('xyz')= 3</code> , <code>len('编程') = 2</code> |
| str (x) | 返回任意类型x对应字符串，如 <code>str(3.14159)= '3.14159'</code> |
| chr (x) | 返回Unicode编码x对应字符，如 <code>chr(97) = 'a'</code> , <code>chr(65) = 'A'</code> |
| ord (x) | 返回字符x对用Unicode编码，如 <code>ord('a') = 97</code> , <code>ord('A') = 65</code> |
| hex (x) | 返回整数x对应对应十六进制数的小写形式字符串，如 <code>hex(255) = '0xff'</code> |
| oct (x) | 返回整数x对应对应八进制数的小写形式字符串，如 <code>oct(255) = '0o377'</code> |



3.5 字符串类型及其操作



☁ 内置字符串处理函数

☁ 微实例：凯撒密码

明文字母表：

ABCDEFGHIJKLMNOPQRSTUVWXYZ

密文字母表：

DEFGHIJKLMNOPQRSTUVWXYZABC

加密：密文 = (明文 + 3) mod 26

解密：明文 = (密文 - 3) mod 26

#3.2CaesarCode.py

```
plaincode = "python is an excellent language"
```

```
for p in plaincode:
```

```
    if ord("a") <= ord(p) <= ord("z"): # 仅支持小写英文字母
```

```
        print(chr(ord("a")+(ord(p)-ord("a")+3)%26), end='')
```

```
    else:
```

```
        print(p, end='')
```

(运行结果)

```
sbwkrq lv dq hafhoohqw odqjxdjh
```




3.5 字符串类型及其操作



☞ 内置字符串处理方法

☞ 43种内置字符串处理方法，常用16种

| 方 法 | 描 述 |
|-------------------------|--|
| str.lower() | 返回字符串str的小写形式，如 <code>'XYZ'.lower() = 'xyz'</code> |
| str.upper() | 返回字符串str的小写形式，如 <code>'xyz'.upper() = 'XYZ'</code> |
| str.islower() | 当str所有字符都是小写时返回True，如 <code>'xy'.islower()=True</code> |
| str.isnumeric() | 当str所有字符都是数字时返回True，如 <code>'12'.isnumeric()=True</code> |
| str.isspace() | 当str所有字符都是小写时返回True，如 <code>' '.isspace() = True</code> |
| str.split(x) | 根据字符x把str分割为列表，如 <code>'a,bc,d'.split(',') = ['a', 'bc', 'd']</code> |
| str.count(x) | str中子串x出现次数，如 <code>'abcabb'.split('ab') = 2</code> |
| str.replace(x,y) | str中子串x替换为y后返回，如 <code>'xya'.replace('a', 'z') = 'xyz'</code> |
| str.strip() | 去除str中两端的空格（换行符），如 <code>' y \n'.strip() = 'y'</code> |
| str.join(list) | 将list中的元素通过str连接为字符串，如 <code>','.join(['a','b']) = 'a,b'</code> |



3.6 字符串类型的格式化



3.6 字符串类型的格式化



❖ 格式化方式

'小明: 男, 19岁'

❖ 格式化符号% (占位符) 类似于C语言, 不够Pythonic

✓ '%s %d %f' % ('str', 123, 3.14)

✓ '%(name)s: %(age)d' % {'name': 'Tom', 'age': 19}

❖ 槽格式 ({}) 和format()方法

✓ '{} {} {}'.format('str', 123, 3.14)

✓ '{0} {1} {2}'.format('str', 123, 3.14)

✓ '{name}: {age}'.format(name='Tom', age=19)

❖ 格式化字符串常量f-string Python3.6及以上版本

✓ f'{name}: {age}' f'{name.lower()}'

✓ f'{2 * 37}' f'{a * b}'



3.6 字符串类型的格式化



🔗 **format()方法** 返回一个新的字符串

🔗 基本使用格式

<模板字符串>.**format**(<逗号分割的参数>)

🔗 模板字符串由槽组成，槽用大括号**{}**表示

🔗 若{}没有序号，则按出现顺序依次替换

✓ '{ } { } { }'.**format**('str', 123, 3.14)

🔗 若{}有序号，则按序号对应参数替换，序号从0开始

✓ '{2} {1} {0}'.**format**('str', 123, 3.14)

🔗 通过字典设置参数

✓ '{name}: {age}'.**format**(name='Tom', age=19)



3.6 字符串类型的格式化



❧ format()方法的格式控制

❧ 槽内部控制样式

{<参数序号>:<格式控制标记>}

❧ 格式控制标记内容 如"{:-^30s}" "{:-^20,.2f}"

- ❧ 填充字符：如----python----(默认为空格)
- ❧ 对齐方式：<(左对齐), >(右对齐), ^(居中对齐)
- ❧ 输出宽度<N>： 如python (默认为字符串本身长度)
- ❧ 数字千分位分隔符<,>： 如12,345,678
- ❧ 浮点数小数部分的精度<.N>： 如3.14
- ❧ 数据类型：s (字符串), d(十进制整数), f(浮点数), e(指数形式), %(百分数形式)



3.6 字符串类型的格式化



format()方法的格式控制实例

#字符串格式化

```
"{:30s}".format("Python")
"{:30}".format("Python")
'Python'
"{:>30}".format("Python")
'Python'
"{:*^30}".format("Python")
'*****Python*****'
"{:.4}".format("Python")
'Pyth'
```

#整数格式化

```
"{:20,d}".format(1234567890)
'1,234,567,890'
"{: ^20,}".format(1234567890)
'---1,234,567,890---
```

#浮点数格式化

```
"{: ^20,f}".format(12345.678)
'---12,345.678000---'
"{: ^20,.2f}".format(12345.678)
'-----12,345.68-----'
"{: ^20,.2e}".format(12345.678)
'-----1.23e+04-----'
"{: ^20,.2%}".format(0.6789)
'-----67.89%-----'
```



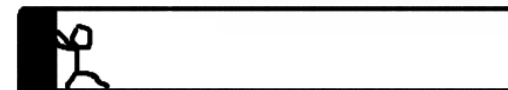
3.7 实例：文本进度条



3.7 实例：文本进度条



进度条



- 用于计算机处理任务或执行软件中增强用户体验
- print()函数实现文本进度条
 - 多行非刷新显示
 - 单行动态刷新显示

```
% 0
% 10
% 20
% 30
% 40
% 50
% 60
% 70
% 80
% 90
% 100
```

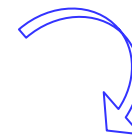


```
% 100
```



百分比

```
% 0 [->.....]
% 10 [**->.....]
% 20 [***->.....]
% 30 [*****->.....]
% 40 [*****->.....]
% 50 [*****->.....]
% 60 [*****->.....]
% 70 [*****->.....]
% 80 [*****->.....]
% 90 [*****->.....]
% 100 [*****->.....]
```



```
% 100 [*****->]
```



百分比+进度条



3.7 实例：文本进度条



百分比多行非刷新显示

```
#3.3.TextProgressBar.py
import time
scale = 10
#方式1：百分比多行非刷新显示
print("方式1：百分比多行非刷新显示")
print("-----执行开始-----")
for i in range(scale+1):
    c = (i / scale) * 100
    print("%{:3.0f}".format(c))
    time.sleep(0.1)
print("-----执行结束-----")
```

`time.sleep(t)`

将当前程序暂时挂起t秒



运行示例



3.7 实例：文本进度条



百分比单行动态刷新显示

```
#3.3.TextProgressBar.py
import time
scale = 10
#方式2：百分比单行动态刷新显示
print("方式2：百分比单行动态刷新显示")
print("-----执行开始-----")
for i in range(scale+1):
    c = (i / scale) * 100
    print("\r%{:3.0f}".format(c), end='')
    time.sleep(0.1)
print()
print("-----执行结束-----")
```



运行示例

```
print("\r%", end='')
```

光标回到行首

print()不换行



3.7 实例：文本进度条



百分比进度条多行非刷新显示

#3.3.TextProgressBar.py

```
import time
```

```
scale = 10
```

```
#方式3：百分比进度条多行非刷新显示
```

```
print("方式3：百分比进度条多行非刷新显示")
```

```
print("-----执行开始-----")
```

```
for i in range(scale+1):
```

```
    a, b = "*" * i, '..' * (scale - i)
```

```
    c = (i / scale) * 100
```

```
    print("%{:3.0f} [{}->{}]" .format(c, a, b))
```

```
    time.sleep(0.1)
```

```
print("-----执行结束-----")
```

逐渐增加前端 *，逐渐减少后端 .

```
% 0 [->.....]  
%100 [*****->]
```

运行示例



3.7 实例：文本进度条



百分比进度条单行动态刷新显示

#3.3.TextProgressBar.py

```
import time
```

```
scale = 10
```

```
#方式4：百分比进度条单行动态刷新显示
```

```
print("方式2：百分比单行动态刷新显示")
```

```
print("-----执行开始-----")
```

```
for i in range(scale+1):
```

```
    a, b = "***" * i, '..' * (scale - i)
```

```
    c = (i / scale) * 100
```

```
    print("\r%{:3.0f} [{}->{}]".format(c, a,
```

```
b), end='')
```

```
    time.sleep(0.1)
```

```
print()
```

```
print("-----执行结束-----")
```



运行示例



3.7 实例：文本进度条



高精度进度条单行动态刷新+时间显示

#3.4.TextProgressBar.v2.py

```
import time
scale = 50
print("{:-^{}}".format("执行开始", scale//2))
t = time.clock()
for i in range(scale+1):
    a = "*" * i
    b = '.' * (scale - i)
    c = (i / scale) * 100
    t -= time.clock()
    print("\r{3.0f}%[{}->{}] {:.2f}s".format(c, a, b, abs(t)),
end='')
    time.sleep(0.1)
print()
print("{:-^{}}".format("执行结束", scale//2))
```

`time.clock()`

获取当前时间



 <https://github.com/tqdm/tqdm>

```
from tqdm import tqdm
for i in tqdm(range(10000)):
    ...
```

[illegible]

tqdm使用示例



本章要点



Python语言的基本数据类型

数字

- 3种类型
- 数值运算操作符
- 数值运算函数
- 数值类型转换函数

字符串

- 字符串的表示
- 字符串操作符、处理函数和方法
- 字符串format()格式化方法



程序练习题



3.1 重量计算

月球上物体重量为在地球上的16.5%，假如某人在地球上每年增长0.5kg，编写程序输出该人未来10年在地球上和月球上的体重情况。

代码文件名: 3.1EarthMoonWeight.py

3.2 回文数判断

如果一个自然数 n （如1234321）的各位数字反向排列所得数字与 n 相等，则 n 为回文数。编写程序，从键盘输入一个5位数字，判断该数字是不是回文数。

代码文件名: 3.2PalindromeNumber.py



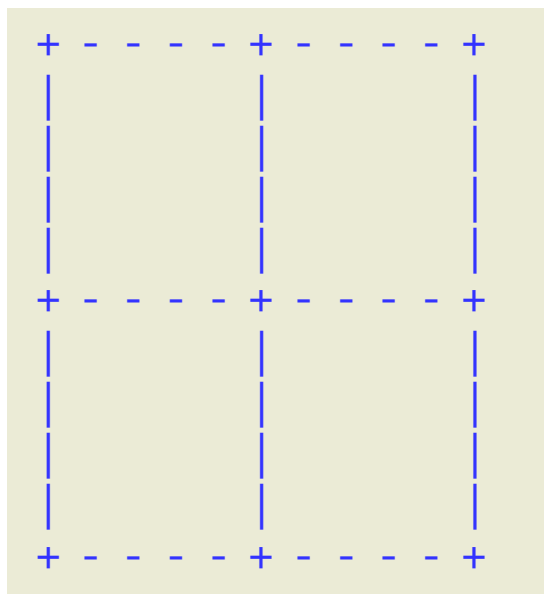
程序练习题



3.3 输出田字格

使用print函数打印出如图所示田字格。

代码文件名：3.3PrintMatts.py



全角字符
"+", "-", "|", " "

.py代码文件打包(3.学号+姓名)发送到
python_xxmu@163.com



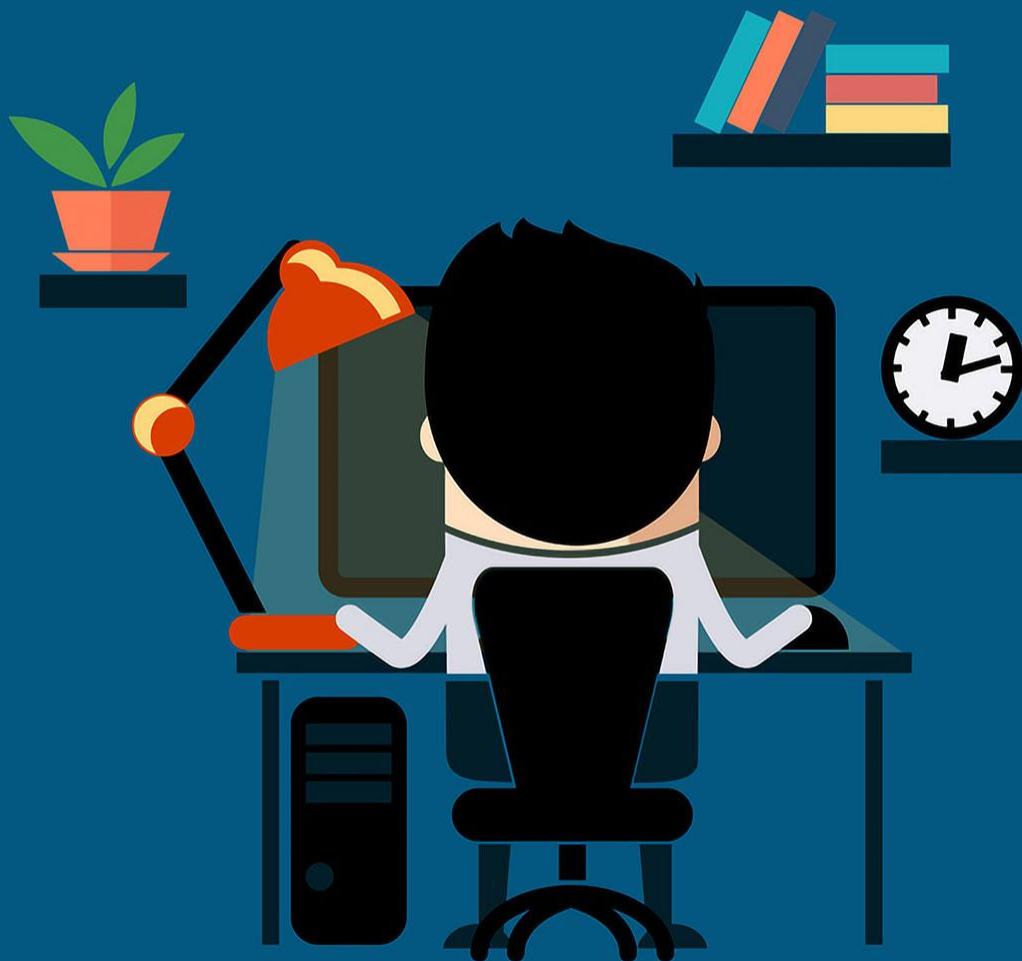
下周课程



🔗 第4章 Python程序控制结构

- 🔗 程序的基本结构
- 🔗 程序的分支结构
- 🔗 实例：身体质量指数BMI
- 🔗 程序的循环结构
- 🔗 模块：random库的使用
- 🔗 程序的异常处理

编程辣么好，还等什么？开始学习吧！



Programing is an Art