



# 《人工智能与Python程序设计》——基本数据类型



人工智能与Python程序设计 教研组



# 上次课回顾：初识Python

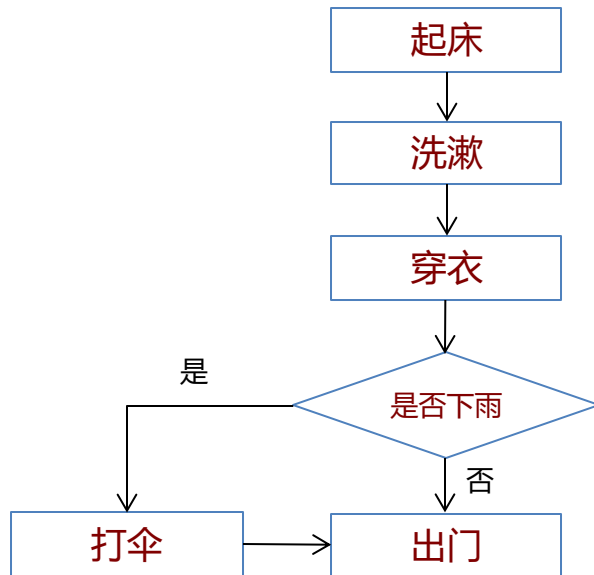
- 什么是程序

- 严格的表示过程性知识的方式

- 陈述性知识 (declarative knowledge) vs 过程性知识 (procedure knowledge)

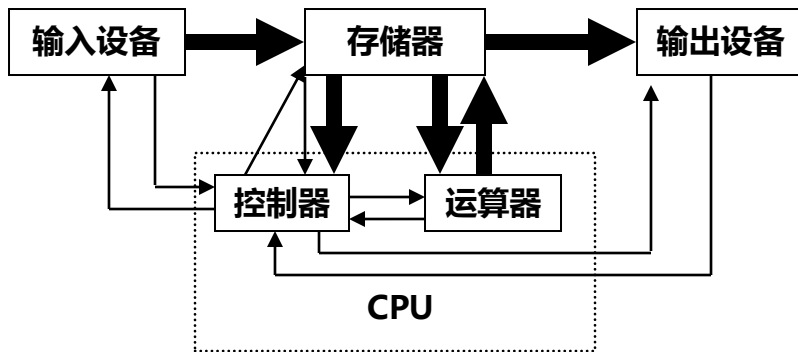
- 计算机程序

- 为了完成某项任务，解决某个问题由计算机执行的一系列指令（步骤）
  - 通常是按照程序指定的过程对特定的数据进行处理和计算



# 上次课回顾：初识Python

- 冯·诺依曼体系结构



- 计算机之父：冯·诺依曼

- 计算机应该按照程序顺序执行
- 采用二进制作作为计算机的数制基础
- 程序也应该被保存在存储器（和数据使用同一个存储器）
- 《First Draft of a Report on the EDVAC》1945年6月30日

- 第二届图灵奖得主：莫里斯·威尔克斯

- 设计和**制造**第一台内部存储程序的电子计算机EDSAC
- 第一台“冯·诺依曼体系结构”的计算机

- 谁首先提出存储程序这一想法难以考证





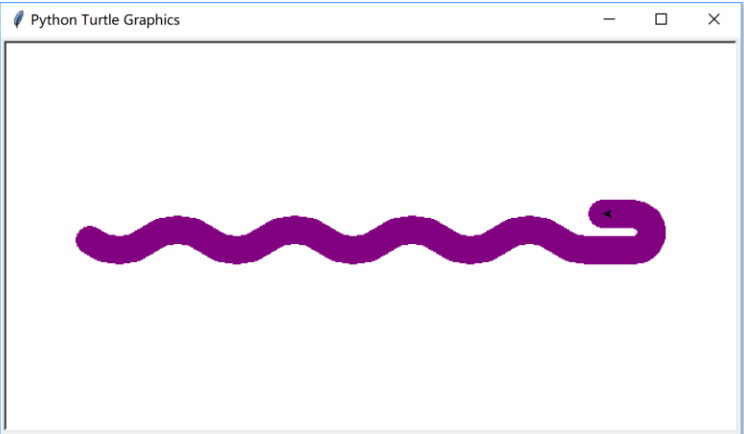
# 上次课回顾：初识Python

- 设计一个Python程序
  - 问题分析与模型
  - 程序格式与框架
  - 变量命名规则
  - 字符串
  - 注释
  - Python保留字
  - 输入/输出函数
  - 流程控制语句

```
#TempConvert.py
TempStr = input("请输入带有符号的温度值：")
if TempStr[-1] in ['F', 'f']:
    C = (eval(TempStr[0:-1]) - 32)/1.8
    print("转换后的温度是{:.2f}C".format(C))
elif TempStr[-1] in ['C', 'c']:
    F = 1.8*eval(TempStr[0:-1]) + 32
    print("转换后的温度是{:.2f}F".format(F))
else:
    print("输入格式错误")
```

# 上次课回顾：初识Python

- 运行和调试Python程序
  - IDE
  - Jupyter notebook
  - 命令行/Terminal

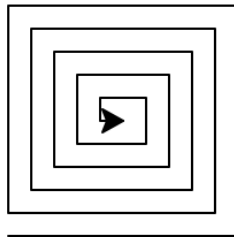
A screenshot of a Python Turtle Graphics window. The window title is "Python Turtle Graphics". It displays a purple wavy line drawn on a white background. The line starts with a small circle on the right and moves left, creating a series of connected semi-circles.

```
#PythonDraw.py
import turtle
turtle.setup(650 , 350, 200, 200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
turtle.seth(-40)
for i in range(4):
    turtle.circle(40,80)
    turtle.circle(-40, 80)

turtle.circle(40, 80/2)
turtle.fd(40)
turtle.circle(16,180)
turtle.fd(40 * 2/3)
turtle.done()
```

# 课后练习

- 修改绘制蟒蛇程序的参数，观察修改的效果
- 用turtle绘制正方形螺旋线
  - 用一个IDE编写
  - 如果程序运行出错
    - 仔细阅读出错信息，分析并修改你的程序
  - 如果执行效果与期待有差异
    - 通过单步执行观察每一行程序的执行效果是否符合你的设计，找出分析差异，修改你的程序





# 上次课回顾：初识Python

- Python、Anaconda、PyCharm、VS code、Jupyter之间的关系
  - Python
    - 一种程序设计语言
    - 一个可以解释执行Python语言编写的程序的程序
      - cpython: 用C语言实现, 最常见的python解释器, python语言的reference implementation
      - 其他python解释器: Jython (Java语言实现的python解释器)、pypy (python语言实现的python解释器)
      - **注意: 你的电脑上可能安装了不只一个python解释器**
  - Anaconda
    - Python的一个用于数据分析、科学计算、人工智能的发行版本
    - 致力于简化软件包管理系统和部署, 包管理程序conda
    - Anaconda自带了Jupyter



# 上次课回顾：初识Python

- Python、Anaconda、PyCharm、VS code、Jupyter之间的关系
  - PyCharm/VS code
    - 集成开发环境（IDE）：帮助程序员高效开发软件代码的软件应用程序
    - 将软件编辑、构建、测试和打包等功能结合到一个应用程序中
    - 你可以在PyCharm/VS code里使用jupyter
  - Jupyter notebook/lab
    - 一个基于Web的**交互式 (interactive)** 计算平台
    - 支持在线运行Python、R等语言编写的代码
    - 显示描述性文字、公式、图片、可视化结果
    - 适合用于教学和分享数据分析的代码、过程、结果





# 教学目标

- 了解和掌握Python语言提供的数字类型
  - 整数、浮点数、复数
- 掌握使用Python进行数字运算
  - 二元运算、增强赋值语句、math库
- 掌握关系运算符和逻辑运算符
  - 布尔类型
- 了解和掌握Python语言提供的字符串类型
  - 字符串的+和\*运算
  - 字符串的格式化处理 (format()方法)
  - 输出进度条



# 提纲



## Python 02 基本数据类型

- ☐ 数字及基本操作
- ☐ 字符串及其操作
- 
-



# Python中的数字

- 程序元素：010 / 10，存在多种可能
  - 表示十进制整数值10
  - 字符串
- 数字类型对Python语言中数字的表示和使用进行了定义和规范
- Python中的数字类型包括
  - 整数
  - 浮点数
  - 复数



# 整数

- 与数学中的整数概念一致，没有取值范围限制
- 数的运算： `pow(x, y)`：计算x的y次方
- 课堂练习：在Jupyter Notebook中计算
  - `pow(2, 10)`, `pow(10, 2)`
  - `pow(2, 1000)`
  - `pow(2, pow(2, 3))`

# 其它整数表示的例子

- 负数: -2010, -100
- 16进制整数表示: 以0x或者0X开头
  - Hexadecimal
  - 0x11、0x1a
- 2进制整数表示: 以0b或者0B开头
  - Binary
- 8进制整数表示: 以0o或者0O开头
  - Octal
- 10进制 (Decimal)

```
▶ a = 0x11  
print(a)
```

17

```
▶ print(0b11)  
print(0011)
```

3

9

# 浮点数的表示形式

- 带有小数点及小数位的数
- 最常用：小数
  - 0.12、-77.、-2.17
- 科学计数法：使用字母 “e” 或者 “E” 作为幂的符号，以10为基数
  - 科学计数法含义：  $a E b = a * 10^b$
  - 96e4、4.3e-1、9.6E5

```
▶ i = 4.3e2  
j = 4.3 * pow(10, 2)  
print(i, j)
```

430.0 430.0



# 浮点数

- Python语言中对浮点数的表示存在限制
  - 数值范围存在限制（上溢出，overflow）
  - 小数精度也存在限制（下溢出，underflow）
  - 这种限制与在不同计算机系统有关

```
i = pow(2, 5000)
print("i = ", i)

j = i/3.3
print("j = ", j)
```

```
i = 141246703213942603683520966701614733366889617518454111681368808585711816984270751255808912631671152637335603208431366082764203838069
9793383359711857266399234310517778518653990118779996451317070693734982126313237525531121537284403595090053595486073341845340557556673680
15655874054646996404990508496994723579009056175713766182282164342131815209915566771264986517822041740618309392391768613413832940182402258
38692725596147005144243281075275629495339093813198966735633606329691023842454125835888656873133981287240980008838073668221804264432910894
03078902021944057819848826733976823887227990215742030724757051042384586887259673589180581872779643575301851808664135601285130254672682300
92502183280182519073402454498631832656379878621985110463629854619495872811191399072280043859428809539588165545676252960869168857748289344
4994136241658867532694033256110366456982622206834474219811081872404929503481991376740379825998791411879802717583885498575115299471743469
24111707023039810337861523279371029099265644484289551183035573315202080415792009004181195188045670551546834944618273174232768598927760762
07095258783187664883683489650154749978641197654414333569280123441117657353363935578792149370043475682086659587177640592935928875142928435
57047089164876483116615691886203812997556690171892169733755224469032475078797830901321579940127337210694377283439922280274060798234786740
434893458120196341110103381250672004660989116070028400210098045296403978870433530261933759786205219228037148113216414718651416909091719190
9376
```

```
OverflowError                                Traceback (most recent call last)
<ipython-input-13-f3257e3833b4> in <module>
      2 print("i = ", i)
      3
--> 4 j = i/3.3
      5 print("j = ", j)
```

OverflowError: int too large to convert to float

```
▶ i = 1.0 + pow(2, -5000)
print("i = ", i)
```

i = 1.0

# 复数

- 与数学中复数的概念一致
  - $a + bj$ ,  $j = \sqrt{-1}$
  - $a$ : 实部,  $b$ : 虚部
- 如  $z = 1 + 5j$ 
  - `z.real` : 获取实部
  - `z.imag`: 获取虚部
- 复数的运算规则
  - $z_1 = a + bj$ ,  $z_2 = c + dj$  是任意两个复数
  - 和:  $(a + bj) + (c + dj) = (a + c) + (b + d)j$
  - 差:  $(a + bj) - (c + dj) = (a - c) + (b - d)j$
  - 积:  $(a + bj)(c + dj) = (ac - bd) + (bc + ad)j$

```
z = 1.0 + 1.0j  
print(z.real, z.imag)
```

```
z **= 2  
print(z.real, z.imag)
```

1.0 1.0

0.0 2.0



# 数字类型的操作

- 不同数字类型之间可以进行混合运算，运算后生成结果为最宽类型
  - $123+4.0=127.0$  (整数+浮点数=浮点数)
- 规则（按照最宽的返回）：
  - 整数之间运算，如果结果一定是整数（包括整除、求余），结果是整数；
  - 整数之间的**除法**（“/” 操作符），结果是浮点数；
  - 整数和浮点数混合运算，输出结果是浮点数

操作符	描述
$x + y$	x与y之和
$x - y$	x与y之差
$x * y$	x与y之积
$x / y$	x与y之商
$x // y$	x与y之整数商，即：不大于x与y之商的最大整数
$x \% y$	x与y之商的余数，也称为模运算
$-x$	x的负值，即： $x*(-1)$
$+x$	x本身
$x**y$	x的y次幂，即： $x^y$



# 数值的类型判断与转换

- 数值类型判断: `type(x)`
- 方式1: 数值运算操作符可以隐式地转换输出结果的数字类型
- 方式2: 通过内置的数字类型转换函数可以显式地在数字类型之间进行转换
  - `float(5)=5.0` (增加小数部分)
  - `int(5.6)`返回5 (直接去掉小数部分, 有精度损失)

```
▶ i = 5
print(type(i))

i = i + 0.5
print(i)
print(type(i))

<class 'int'>
5.5
<class 'float'>
```

```
▶ i = 5
print(type(i))
i = float(i)
print(i)
print(type(i))

print()
i = i + 0.6
i = int(i)
print(i)
print(type(i))

<class 'int'>
5.0
<class 'float'>

5
<class 'int'>
```



# 二元操作符号对应的增强赋值操作符

- $x = x \text{ op } y \Leftrightarrow x \text{ op} = y$ 
  - 其中op为二元操作符
  - 例如: op可以为 + - \* / \*\* %等

$x \text{ +} = y$      $x \text{ -} = y$      $x \text{ *} = y$      $x \text{ /} = y$

$x \text{ //} = y$      $x \text{ \%} = y$      $x \text{ **} = y$

- 举例:  $x = 3.14$ 
  - $x = x \text{ ** } 3 \Leftrightarrow x \text{ **} = 3$

```
| x = 3.14
  x = x ** 3
  print (x)

x = 3.14
x **= 3
print (x)
```

30.959144000000002

30.959144000000002



# 数值运算函数

函数及使用	描述
<code>abs(x)</code>	绝对值，x的绝对值 <code>abs(-10.01)</code> 结果为 10.01
<code>divmod(x,y)</code>	商余， $(x//y, x\%y)$ ，同时输出商和余数 <code>divmod(10, 3)</code> 结果为 (3, 1)
<code>pow(x, y[, z])</code>	幂余， $(x**y)\%z$ ，[...]表示参数z可省略 <code>pow(3, pow(3, 99), 10000)</code> 结果为 4587
<code>round(x[, d])</code>	四舍五入，d是保留小数位数，默认值为0 <code>round(-10.123, 2)</code> 结果为 -10.12
<code>max(x<sub>1</sub>, x<sub>2</sub>, ..., x<sub>n</sub>)</code>	最大值，返回 $x_1, x_2, \dots, x_n$ 中的最大值，n不限 <code>max(1, 9, 5, 4, 3)</code> 结果为 9
<code>min(x<sub>1</sub>, x<sub>2</sub>, ..., x<sub>n</sub>)</code>	最小值，返回 $x_1, x_2, \dots, x_n$ 中的最小值，n不限 <code>min(1, 9, 5, 4, 3)</code> 结果为 1
<code>int(x)</code>	将x变成整数，舍弃小数部分 <code>int(123.45)</code> 结果为123； <code>int("123")</code> 结果为123
<code>float(x)</code>	将x变成浮点数，增加小数部分 <code>float(12)</code> 结果为12.0； <code>float("1.23")</code> 结果为1.23
<code>complex(x)</code>	将x变成复数，增加虚数部分 <code>complex(4)</code> 结果为 4 + 0j



# math库

- Python语言本身提供的数值运算符号有限，是Python提供的内置数学类函数库
  - 提供了4个数学常数和44个函数
  - pi、e、无穷大、非数值
  - 16个数值表示函数
  - 8个幂对数函数
  - 16个三角对数函数
  - 4个高等特殊函数
- 使用方法
  - 1. 引入math库: `import math`
  - 2. 调用math库中的功能函数，如: `math.cos(math.pi)`
- 注意：计算机无法直接表示无理数，用float近似表达，有精度损失

```
import math
x = 2
y = math.sqrt(2)
print(y)
print(type(y))
```

```
1.4142135623730951
<class 'float'>
```

```
print(math.pi)
y = math.cos(math.pi)
print(y)
z = math.cos(math.pi / 3)
print(z)
```

```
3.141592653589793
-1.0
0.5000000000000001
```



# math库中的函数



函数	数学表示	描述
math.fabs(x)	$ x $	返回x的绝对值
math.fmod(x, y)	$x \% y$	返回x与y的模
math.fsum([x,y,...])	$x+y+\dots$	浮点数精确求和
math.ceil(x)	$\lceil x \rceil$	向上取整, 返回不小于x的最小整数
math.floor(x)	$\lfloor x \rfloor$	向下取整, 返回不大于x的最大整数
math.factorial(x)	$x!$	返回x的阶乘, 如果x是小数或负数, 返回ValueError
math.gcd(a, b)		返回a与b的最大公约数
math.frexp(x)	$x = m * 2^e$	返回(m, e), 当x=0, 返回(0.0, 0)
math.ldexp(x, i)	$x * 2^i$	返回 $x * 2^i$ 运算值, math.frexp(x)函数的反运算
math.modf(x)		返回x的小数和整数部分
math.trunc(x)		返回x的整数部分
math.copysign(x, y)	$ x  *  y /y$	用数值y的正负号替换数值x的正负号
math.isclose(a,b)		比较a和b的相似性, 返回True或False
math.isfinite(x)		当x为无穷大, 返回True; 否则, 返回False
math.isinf(x)		当x为正数或负数无穷大, 返回True; 否则, 返回False
math.isnan(x)		当x是NaN, 返回True; 否则, 返回False

函数	数学表示	描述
math.pow(x,y)	$x^y$	返回x的y次幂
math.exp(x)	$e^x$	返回e的x次幂, e是自然对数
math.expml(x)	$e^x - 1$	返回e的x次幂减1
math.sqrt(x)	$\sqrt{x}$	返回x的平方根
math.log(x[,base])	$\log_{base} x$	返回x的对数值, 只输入x时, 返回自然对数, 即 $\ln x$
math.log1p(x)	$\ln(1 + x)$	返回1+x的自然对数值
math.log2(x)	$\log x$	返回x的2对数值
math.log10(x)	$\log_{10} x$	返回x的10对数值



# math库中的函数

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)	$\operatorname{arsinh} x$	返回x的反双曲正弦函数值
math.acosh(x)	$\operatorname{arcosh} x$	返回x的反双曲余弦函数值
math.atanh(x)	$\operatorname{artanh} x$	返回x的反双曲正切函数值

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， $\text{math.erfc}(x)=1 - \text{math.erf}(x)$
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)	$\ln(\text{gamma}(x))$	伽玛函数的自然对数



# 示例：DayDayUp365

- 问题：一年365天，如果好好学习时能力值相比前一天提高1‰，当放任时相比前一天下降1‰。效果相差多少呢？

```
▶ import math
ratio = 0.001
dayup = math.pow(1 + ratio, 365) #day day up
daydown = math.pow(1 - ratio, 365) # day day down
print("向上: {:.2f}, 向下: {:.2f}".format(dayup, daydown))
```

向上: 1.44, 向下: 0.69

- 课堂练习：输入上述程序，试试提高ratio（比如：5‰、1%），感受一下指数的力量





# 关系运算符 (Relational Operators)

- 判断一个数  $x$  是否为偶数
  - $x \% 2$  是否等于 0
  - $x \% 2 == 0$
  - 若为True, 则  $x$  为偶数
  - 若为False, 则  $x$  为奇数
- 用于判断两个值的关系
  - 大小、相等或不相等
- 运算的结果只有两种 (布尔型)
  - 若结果为True, 表示条件成立
  - 若结果为False, 表示条件不成立

关系运算符	含义	举例
==	等于 (equal)	10 == 20 is False
!= , <>	不等于 (not equal)	10 != 20 is True
>	大于 (greater)	10 > 20 is False
<	小于 (less)	10 < 20 is True
>=	大于等于 (greater or equal)	10 >= 20 is False
<=	小于等于 (less or equal)	10 <= 20 is True

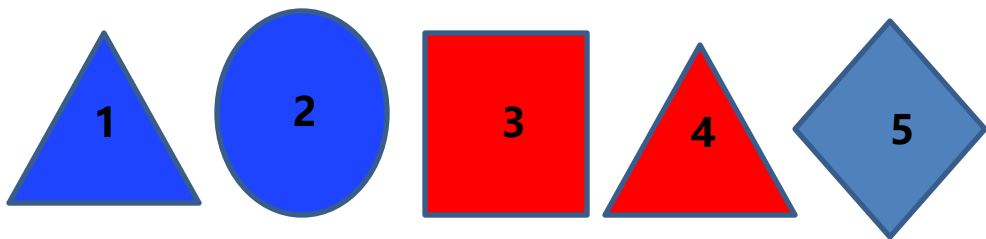


# 逻辑运算符 (Logical Operators)

- 现实世界中处处体现逻辑
  - 你们班有没有身高一米九以上的男生? 身高 > 1.9 and 性别 == 男
  - 地铁里禁止喝水吃东西 禁止: 喝水 or 吃东西
  - .....
- 逻辑运算符

关系运算符	含义	举例
and	与 (全真才真)	True and False == False
or	或 (全假才假)	True or False == True
not	非 (真变假、假变真)	not True == False

# 逻辑运算示例



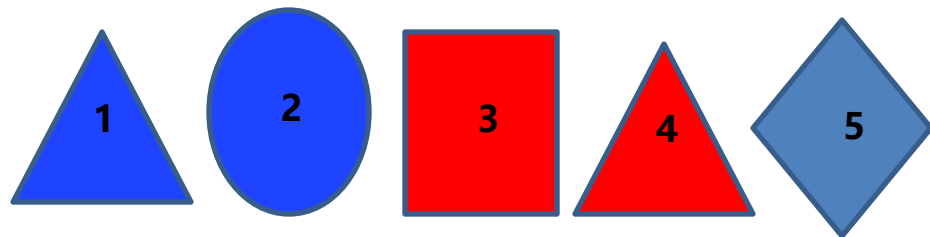
该图形是否为红色三角形？

颜色 == 红色      and      形状 == 三角形

	A	B	A and B
1	F	T	F
2	F	F	F
3	T	F	F
4	T	T	T
5	F	F	F

<i>A</i>	<i>B</i>	<i>A and B</i>
F	F	F
F	T	F
T	F	F
T	T	T

# 逻辑运算示例



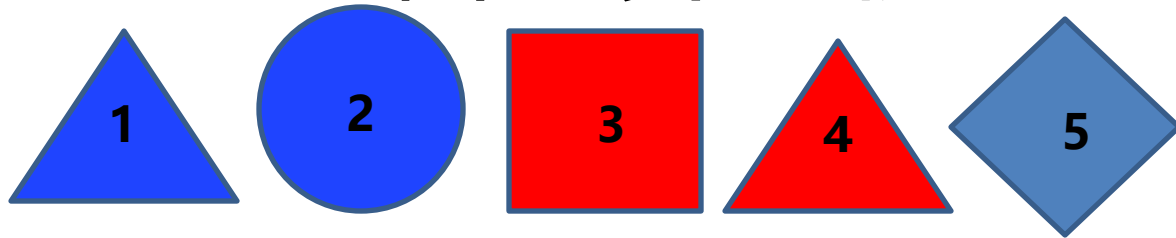
该图形是否为红色或三角形？

颜色 == 红色 or 形状 == 三角形

	A	B	A or B
1	F	T	T
2	F	F	F
3	T	F	T
4	T	T	T
5	F	F	F

<i>A</i>	<i>B</i>	<i>A or B</i>
F	F	F
F	T	T
T	F	T
T	T	T

# 逻辑运算示例



该图形是否非红色？

not 颜色 == 红色

	A	not A
1	F	T
2	F	T
3	T	F
4	T	F
5	F	T

<i>A</i>	<i>not A</i>
T	F
F	T

# 逻辑运算示例：判断闰年

- 如果年份  $y$  能被 4 整除但是不能被 100 整除，或者能被 400 整除，则是闰年
  - 2014、1900 年不是闰年
  - 2012、2000 年是闰年

$(y \% 4 == 0 \text{ and } y \% 100 != 0)$   
or  
 $(y \% 400 == 0)$



▶ #闰年

```
y = int(input("输入年份："))
#年份 y 能被 4 整除但是不能被 100 整除
#或者：能被 400 整除
if (y % 4 == 0 and y % 100 != 0) or (y % 400 == 0):
    print("{0}年是闰年".format(y))
else:
    print("{0}年不是闰年".format(y))
```

输入年份：2000  
2000年是闰年



# 运算符优先级

- 下面两个表达式
  - $2 * 1 + 3$  先乘后加
  - $2 * (1 + 3)$  先加后乘
- 括号  $()$ 
  - 改变了语言内在的默认优先级
  - 具有最高优先级
- 嵌套括号按照由内而外结合
  - $(2 * (1 + 2)) ** 2 == 36$
  - $2 * (1 + 2) ** 2 == 18$

优先级顺序:

- 括号:  $()$
- 幂次:  $**$
- 一元运算:  $+$ ,  $-$
- 算术运算:  $*$ ,  $/$ ,  $\%$ ,  $//$
- 算术运算:  $+$ ,  $-$
- 比较运算:  $==$ ,  $!=$ ,  $<>$ ,  $<=$ ,  $>=$
- 逻辑非:  $not$
- 逻辑与:  $and$
- 逻辑或:  $or$
- 赋值运算:  $=$ ,  $*=$ ,  $/=$ ,  $+=$ ,  $-=$ ,  $\%=$ ,  $//=$

<https://docs.python.org/3/reference/expressions.html#operator-precedence>

$2 ** 2 ** 3$  的结果是 [填空1] 。

- 解决办法：多加括号!!!
  - $2**(2**3)$
  - $(2**2)**3$

正常使用填空题需3.0以上版本雨课堂

作答





# 提纲



## Python 02 基本数据类型

- ☐ 数字及基本操作
- ☐ 字符串及其操作
- 
-



# 字符串类型

- 字符串是用双引号"或者单引号'括起来的一个或多个字符。
- 字符串可以保存在变量中，也可以单独存在
- 可以用type()函数测试一个字符串的类型

- Python语言转义符：
  - 输出带有引号的字符串，可以使用转义符
  - 使用\\输出带有转移符的字符串
  - 使用\n输出带有换行符号的字符串
  - \t

```
str1 = "双引号字符串"
str2 = '单引号字符串'
str3 = '''三引号字符串第一行

三引号字符串第三行'''
print(type(str1), type(str2), type(str3))
print(str1)
print(str2)
print(str3)

print("独立存在的字符串")
```

<class 'str'> <class 'str'> <class 'str'>  
双引号字符串  
单引号字符串  
三引号字符串第一行  
  
三引号字符串第三行  
独立存在的字符串

```
str4 = "带\"的字符串"
str5 = "带\\的字符串"

str6 = '''带"和带\的字符串'''

print(str4)
print(str5)
print(str6)
```

带"的字符串  
带\的字符串  
带"和带\的字符串

# 字符串索引

- 字符串是一个字符**序列**：字符串最左端位置标记为0，依次增加
- 字符串中的编号叫做“索引”

H	e	l	l	o		J	o	h	n
0	1	2	3	4	5	6	7	8	9

- 单个索引辅助访问字符串中的特定位置，如：读取特定位置上的字符
  - Python中字符串索引从0开始，一个长度为L的字符串最后一个字符的位置是L-1
  - Python同时允许使用负数从字符串右边末尾向左边进行反向索引，最右侧索引值是-1

```
▶ greet = "Hello John"
print(greet[1])

x = 8
print(greet[x - 2])

L = len(greet)
print(L, greet[L - 1])
print(greet[-1])

print(greet[-4])
```

```
e
J
10 n
n
J
```



# 字符串的范围索引

- 可以通过两个索引值确定一个位置范围，返回这个范围的**子串**
  - 格式：<string>[<start>:<end>]
  - start和end都是整数型数值，这个子序列从索引start开始直到索引end结束，但不**包括end位置**
- 注意：当<start>位于<end>后面，返回空字符串

```
print(greet)
print(greet[0:3]) #打印位置0-2的子串
print(greet[2:]) #打印从位置2开始后面的子串
print(greet[:4]) #打印位置0-3的子串
print(greet[2:-3]) #前面去掉2个字符，后面去掉3个字符后的字符串

print(greet[3:1])
print(greet[9:-5])
```

```
Hello John
Hel
llo John
Hell
llo J
```

# 字符串运算

- 字符串之间可以通过+或\*进行连接
  - 加法操作(+)将两个字符串连接成为一个新的字符串
  - 乘法操作(\*)生成一个由其本身字符串重复连接而成的字符串
- len: 求字符串长度
  - 英文: 字母数(包括空格)
  - 中文: 汉字数
- str: 把其他类型的数据转为字符串

```
▶ a = "pine"  
b = "apple"  
print(a + b)  
print(3 * a)  
print(a * 3)
```

```
pineapple  
pinepinepine  
pinepinepine
```

```
▶ print(len("abc"))  
print(len("海客谈瀛洲"))
```

```
3  
5
```

```
▶ a = 123  
b = 123e-5  
print(a + b)  
  
str1 = str(123)  
str2 = str(123e-5)  
print(str1+str2)
```

```
123.00123  
1230.00123
```

说明第一行结果123.00123和第二行结果1230.00123是如何计算出来的。

```
▶ a = 123
  b = 123e-5
  print (a + b)

  str1 = str(123)
  str2 = str(123e-5)
  print(str1+str2)
```

```
123. 00123
1230. 00123
```

正常使用主观题需2.0以上版本雨课堂

作答



# 字符串操作

- 常见字符串操作

- 注意：字符串操作并不改变原来的字符串，生产一个新的字符串

- 举例1：替换和变成大写

```
str1 = "Hello John"
str2 = str1.replace("John", "中国人民大学")
str3 = str2.upper()
```

```
print(str1)
print(str2)
print(str3)
```

```
Hello John
Hello 中国人民大学
HELLO 中国人民大学
```

- 举例2：可以通过for和in组成的循环来遍历字符串中每个字符：

for<var>in<string>:  
    操作

操作	含义
+	连接
*	重复
<string>[ ]	索引
<string>[ : ]	剪切
len(<string>)	长度
<string>.upper()	字符串中字母大写
<string>.lower()	字符串中字母小写
<string>.strip()	去两边空格及去指定字符
<string>.split()	按指定字符分割字符串为数组
<string>.join()	连接两个字符串序列
<string>.find()	搜索指定字符串
<string>.replace()	字符串替换
for <var> in <string>	字符串迭代

```
string = "海客谈瀛洲"
for c in string:
    print(c)
```

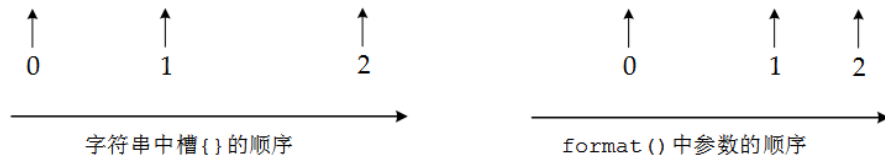
海  
客  
谈  
瀛  
洲



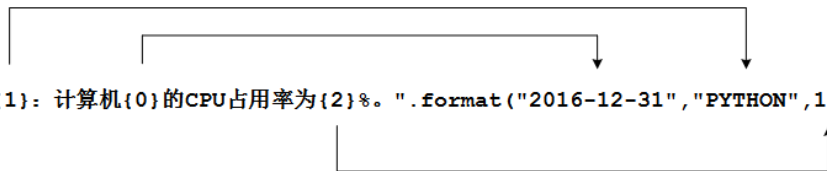
## 字符串格式化处理format()

- 常见于生产格式化的文本，比如：系统日志信息等
- 基本语法格式：<模板字符串>.format(<逗号分隔的参数>)

```
"{ }：计算机{ }的CPU占用率为{ }%。".format("2016-12-31","PYTHON",10)
```



```
"{1}: 计算机{0}的CPU占用率为{2}%.".format("2016-12-31", "PYTHON", 10)
```



```
pattern = "{0}: 计算机{1}的CPU占有率为{2}%。"
string = pattern.format("2020-9-15:12:12:30", "Python", 10)
print(string)
```

2020-9-15:12:12:30: 计算机Python的CPU占有率为10%。



# 更加高级的用法：格式控制

- 槽的内部样式如下：{<参数序号>:<格式控制标记>}
  - 参数序号表示从第几个参数中取值
  - 格式控制标记用来控制参数显示时的格式，包括：
    - <填充><对齐><宽度>,<精度><类型>6个字段
    - 字段都是可选的，可以组合使用

:	<填充>	<对齐>	<宽度>	,	<.精度>	<类型>
引导符号	用于填充的单个字符	< 左对齐 > 右对齐 ^ 居中对齐	槽的设定输出宽度	数字的千位分隔符 适用于整数和浮点数	浮点数小数部分的精度 或字符串的最大输出长度	整数类型 b, c, d, o, x, X 浮点数类型 e, E, f, %

```
pattern = "{0:30}: 计算机{1:*30}的CPU占有率为{2:b}%。"
string = pattern.format("2020-9-15:12:30", "Python", 10)
print(string)
```

2020-9-15:12:30 : 计算机\*\*\*\*\*Python\*\*\*\*\*的CPU占有率为1010%。



# 字符串使用实例1

- 问题：输入一个月份数字，返回对应月份名称缩写
  - 输入：输入一个表示月份的数字(1-12)
  - 处理：利用字符串基本操作实现该功能
  - 输出：输入数字对应月份名称的缩写 (如：1 -> Jan)
- 设计思路
  - 1. 将所有的月份名称缩写存储在一个字符串中

```
months = "JanFebMarAprMayJunJulAugSepOctNovDec"
```
  - 2. 在字符串中截取适当的子串来查找特定月份
  - 3. 找出在哪里切割子串
    - 每个月份的缩写都由3个字母组成
    - 如果pos表示一个月份的第一个字母，则months[pos:pos+3]表示这个月份的缩写
    - monthAbbrev = months[pos:pos+3]



# 字符串使用实例1

- 月份与字符串初始位置关系:  $(n - 1) * 3$

	月份	字符串中位置
Jan	1	0
Feb	2	3
Mar	3	6
Apr	4	9

- 编写程序

```
months = "JanFebMarAprMayJunJulAugSepOctNovDec"
n = int(input("输入月份数(1-12)"))
if n <= 0 or n > 12:
    print("输入1-12的月份数。")
else:
    pos = (n-1)*3
    print(months[pos:pos+3])
```

输入月份数(1-12) 5  
May



## 字符串使用实例2

- 任务：简单的非刷新文本进度条
- 设计思路：
  - 利用print()函数实现
  - 按照任务执行百分比将整个任务划分为100个单位，每执行N%输出一次进度条。
  - 每一行输出包含进度百分比，代表已完成的部分(\*\*)和未完成的部分(..)的两种字符，以及一个跟随完成度前进的小箭头

```
%10 [*****->.....]
```

# 字符串使用实例

- scale=10: 分成10段执行, 每一段提升10%
- 每一个循环画出一个进度条
  - i: 当前的轮数的进度, 0, 1, 2, ..., 10
  - a: 把字符串 “\*\*” 重复i次, 表示完成的进度
  - b: 把字符串 “.” 重复scale - i 次, 表示未完成的进度
  - c: 当前进度百分比, scale = 10时候, 就是 i\*10
  - 通过字符串格式化画得到进度条字符串

```
import time
scale = 10
print("-----执行开始-----")
for i in range(scale + 1):
    a = "*" * i
    b = "." * (scale - i)
    c = (i / scale) * 100
    print("%0:~3.0f}[{1}->{2}]".format(c, a, b))
    time.sleep(0.2)

print("-----执行结束-----")
```

```
-----执行开始-----
% 0 [->.....]
%10 [**->.....]
%20 [***->.....]
%30 [****->.....]
%40 [*****->.....]
%50 [*****->.....]
%60 [*****->.....]
%70 [*****->.....]
%80 [*****->.....]
%90 [*****->.....]
%100 [*****->.....]
-----执行结束-----
```

# 进度条单行动态刷新

- 了解“回车”和“换行”字符
  - 回车<\r>(carriage return): 把打印头定位到左边界 (打印头重新放在这一行的开始)。
  - 换行<\n>(line feed): 把纸向下移一行。
- Unix系统: 每行结尾只有"<换行>", 即"\n";
- Windows: 每行结尾是"<回车><换行>", 即"\r\n";
- Mac: 每行结尾是"<回车>"。
- 如何单行刷新?
  - 只回车, 不换行
  - 用后打印的内容覆盖原来的内容



# 试一下效果

- `print("\r{:3}%".format(i), end="")`
  - `\r`: 回到最左列开始打印
  - `end= ""`: `print`函数不打印`\n`

▶

```
import time
for i in range (101):
    print("\r{:3}%".format(i), end="")
    time.sleep(0.1)
```

100%

- 注意：如果后打印的字符比之前的字符段，不能覆盖的部分将仍然保留在屏幕上

|

```
import time
for i in range (101):
    print("\r{:}%".format(100-i), end="")
    time.sleep(0.1)
```

0%%%



改动图示的程序，实现进度条单行动态刷新

```
import time
scale = 10
print("-----执行开始-----")
for i in range(scale + 1):
    a = "*" * i
    b = "." * (scale - i)
    c = (i / scale) * 100
    print("%0: ^3.0f [{}->{}]".format(c, a, b))
    time.sleep(0.2)

print("-----执行结束-----")
```

```
-----执行开始-----
% 0 [->.....]
%10 [**->.....]
%20 [***->.....]
%30 [****->.....]
%40 [*****->.....]
%50 [*****->.....]
%60 [*****->.....]
%70 [*****->.....]
%80 [*****->.....]
%90 [*****->.....]
%100 [*****->.....]
-----执行结束-----
```

正常使用主观题需2.0以上版本浏览器

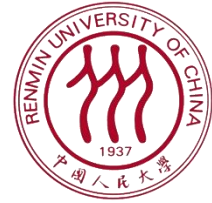
作答





# 练习与思考

- 练习：三天打鱼两天晒网
  - 如果好好学习时能力值相比前一天提高1‰，当放任时相比前一天下降0.8‰
  - 写出程序，计算好好学习3天然后放任2天，学习365天后的能力值
- 思考：关于数的进制
  - 为什么一个整数能被3整除，则它所有数位上数字之和也能被3整除？
  - 还有那些数字具有这样的特点？
  - 8进制和16进制表达的整数，是否也存在这样的数字？



谢谢！