



東南大學
SOUTHEAST UNIVERSITY

人工智能学院

Python编程

授课教师：王洪松

邮箱：hongsongwang@seu.edu.cn

Python编程



- 时间：1-12周，周二(11-12)
- 地点：教二-105，QQ群：1038891178



群名称: Python编程
群 号: 1038891178

考核方式



- 平时成绩 10%
- 期中考试成绩 30% (闭卷)
- 期末考试成绩 60% (闭卷)

本课程特点



- 不是说教，学生 “不待扬鞭自奋蹄”
- 要有C语言基础，或喜欢编程
- 对解决实际问题很有兴趣
- 要求每次课带上电脑，主动编程

不适合的学生（举例）：

- 任何原因缺勤5次及以上
- 发现不带电脑5次及以上
- 想学Python，但不努力练，或没时间练

学习编程有多方面的原因，这些原因可以归结为个人成长、职业发展、创新思维、解决问题能力以及适应未来社会等多个方面。以下是一些主要的原因：

1. **职业发展**：在当今数字化时代，编程技能已成为许多职业领域的核心要求。无论你是希望成为专业的软件工程师、数据科学家、网络安全专家，还是想要在市场营销、教育、医疗等其他领域中利用技术提高效率，编程都是一项非常有价值的技能。掌握编程能让你在就业市场上更具竞争力，甚至开启全新的职业道路。
2. **提升问题解决能力**：编程本质上是一个解决问题的过程。它要求你将复杂的问题分解成更小、更可管理的部分，然后设计并实现解决方案。这种思维方式可以培养你的逻辑思维、批判性思维和创新能力，使你在面对各种挑战时都能找到有效的解决方案。
3. **创新与创造力**：编程是创造新事物的一种方式。通过编程，你可以将自己的想法和创意转化为实际的软件、应用或游戏。这种从无到有的过程可以极大地激发你的创造力和想象力，让你体验到创造的乐趣和成就感。
4. **提高自动化效率**：编程可以帮助你自动化重复性的任务，从而提高工作和生活效率。无论是处理大量数据、优化工作流程，还是创建个性化的服务，编程都能让你以更少的时间和精力完成更多的工作。
5. **适应未来社会**：随着科技的不断发展，人工智能、大数据、物联网等新兴技术正在深刻改变着我们的生活方式和工作环境。掌握编程技能可以让你更好地理解 and 应对这些变化，从而在未来社会中保持竞争力。
6. **跨学科融合**：编程不仅限于计算机科学领域，它已经成为许多学科的交叉点。无论你是学习生物、物理、经济还是其他任何学科，编程都可以帮助你以新的视角和方法来探索和研究问题。
7. **培养耐心和毅力**：编程是一个需要耐心和毅力的过程。在编写代码的过程中，你可能会遇到各种错误和难题，但只有通过不断的尝试、调试和优化，你才能最终完成自己的作品。这个过程可以培养你的耐心和毅力，让你在面对困难和挑战时更加坚韧不拔。

- 如何学习Python编程？
 - 多做编程练习，多和AI大模型对话
 - 主动思考，积极探索，以充分发挥个人的主观能动性
- 本课程要求
 - 上课需要带上笔记本电脑
 - 上课时听讲，关上电脑
 - 讲完课后，打开电脑做编程练习

该如何教python编程



- 有chatgpt后，老师该如何教python编程？
- 教“会用”而不是只教“语法”
- 强调计算思维和问题拆解
- 把 ChatGPT 当成“助教”
- 增强代码理解能力
- 强调实践与项目驱动
- 训练批判性思维



東南大學
SOUTHEAST UNIVERSITY

人工智能学院

Python基础知识

Python 与C++的对比



- 编译型 vs 解释型
 - C++和C一样，也是一种基于编译器的语言。编译器将整个代码转换为特定的操作系统和处理器体系结构的机器语言代码。
 - Python是一种基于解释器的语言。解释器逐行执行源代码。
- 跨平台/可移植性
 - Python代码很容易从一个操作系统移植到另一个操作系统。C++代码不可移植，如果操作系统改变，必须重新编译。
 - Python解释器不会生成编译代码。每次在任何操作系统上运行源代码时，源代码都会转换为字节码，无需任何更改或额外步骤。
- 开发速度
 - C++程序被编译为机器代码。它的执行速度比基于解释器的语言更快。
 - Python解释器不会生成机器代码。每次执行程序时，都会将中间字节码转换为机器语言。

Python 与C++的对比



- 静态类型 vs 动态类型
 - C++是一种静态类型语言。存储数据的变量的类型需要在开始时声明。不能使用未声明的变量。一旦将变量声明为某种类型，只能存储该类型的值。
 - Python是一种动态类型语言。在给变量赋值之前，不需要声明变量。由于变量可以存储任何类型的数据，所以称之为动态类型。
- 垃圾回收
 - C++使用指针的概念。未使用的内存不会自动清除，垃圾回收的过程是手动的。C++程序很经常出现与内存相关的异常行为。
 - Python具有自动垃圾回收机制。不容易出现与内存相关的问题。
- 应用领域
 - C++更适合系统编程、编写设备驱动程序、嵌入式系统和操作系统工具。
 - Python适用于应用程序编程，如数据科学、机器学习、API开发等。

Python 与C++的对比



- 执行效率：C++执行效率高，Python执行效率低。
- 开发效率：C++开发效率低，Python开发效率高。
- 内存管理：python使用自动垃圾收集器进行内存管理，C++程序员必须自己进行内存管理。
- 变量范围：python中通过缩进来表示块，变量范围不限于块；C++通过{}来表示块，变量的范围仅限于由{}划分的这些块。
- 函数调用：如果要调用C++函数，需要在本次调用之前就需要被实现，或者在程序开头事先声明。python没有这个限制，python的函数名也可当作函数参数，函数名也可以当做变量。
- 运算优先级：在运算符和优先级上面，两者并没有大的区别python中没有自加和自减运算符，Python的逻辑运算符是and, or, not, 而C++中则是&&, ||, !

C++和Python在语法的对比



- 认识一门语言，从Hello World!开始

C++版本

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!";
    return 0;
}
```

Python版本

```
#coding=utf-8
print("Hello World!")
```

C++和Python在语法的对比



- 缩进规则

- Python使用缩进来定义代码块，这是Python语法中的一个重要特性。同一代码块内的所有语句必须具有相同的缩进量（空格数），否则会导致语法错误。Python的缩进可以是任意数量的空格（但不建议混合使用空格和制表符），只要保持一致性即可。
- C++则使用大括号{}来定义代码块，缩进主要用于提高代码的可读性，并不是语法的强制要求。

- 动态类型与静态类型

- Python是一种动态类型语言，变量的类型在运行时由解释器根据赋值动态确定。这意味着在编写Python代码时，不需要提前声明变量的类型，也无需在赋值时进行类型转换（除非涉及特殊操作，如将字符串转换为整数）。
- C++是一种静态类型语言，变量的类型在编译时就已经确定。在声明变量时，必须明确指定其类型，并且在后续的使用中，变量的类型不能随意改变（除非使用特定的类型转换操作）。

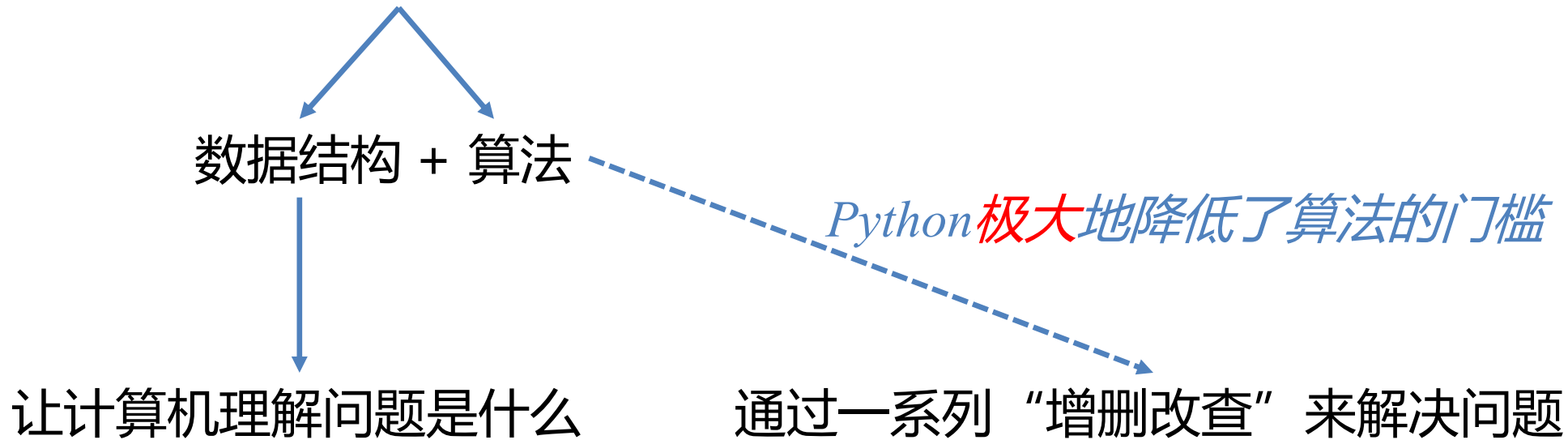
C++和Python在语法的对比



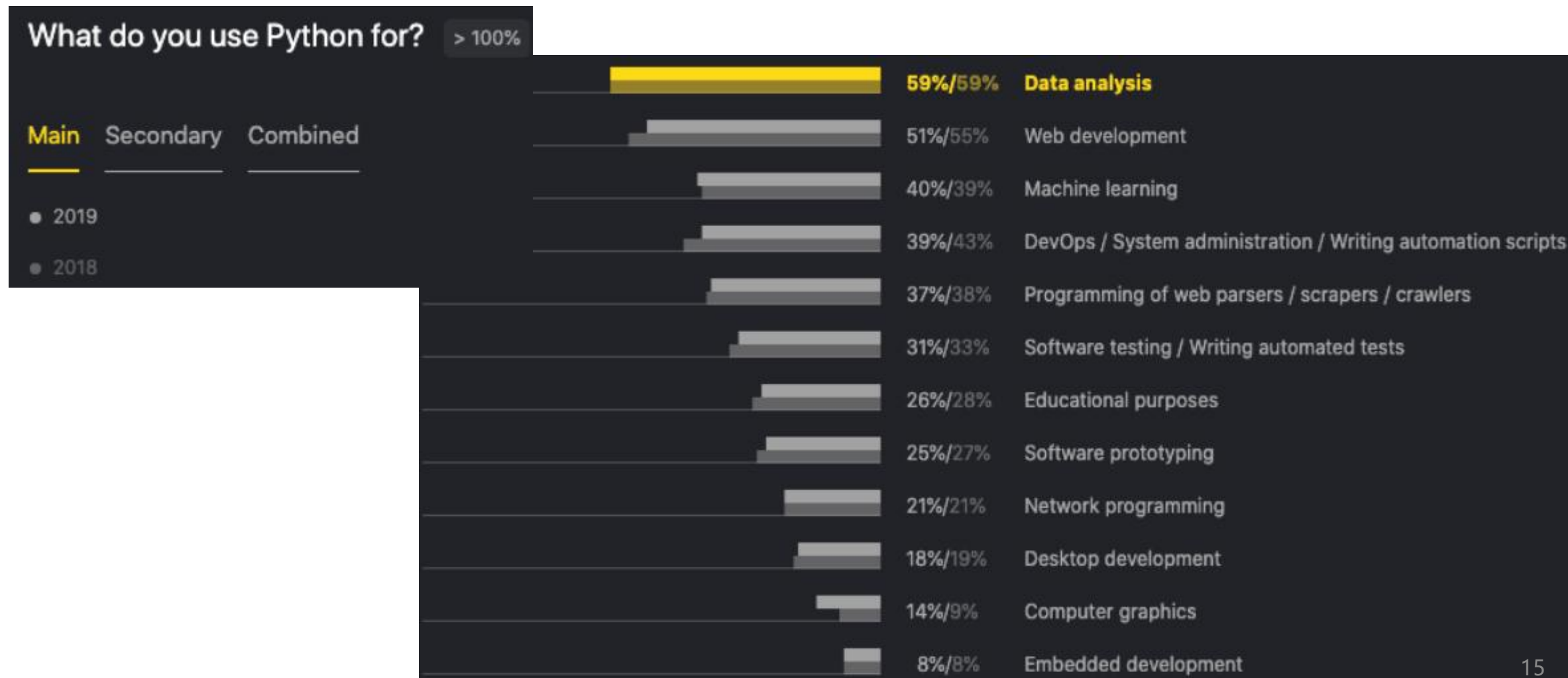
- 变量声明与赋值
 - 在Python中，变量在赋值时自动声明，无需指定类型。例如，`x = 10`会自动将x声明为整型变量。
 - C++中变量需要先声明后使用，且在声明时必须指定类型。例如，`int x = 10;`需要先声明x为整型，然后再赋值。
- 语句终结符
 - Python使用换行符来表示语句的结束，通常不需要在语句末尾添加分号（尽管在某些情况下可以添加分号来在同一行内编写多条语句）。
 - C++中每条语句的末尾都需要一个分号（`;`）来表示语句的结束。
- 模块或库引用
 - Python 使用`import`语句来导入模块或模块中的特定部分。
 - C++ 通过包含头文件（`#include`）来使用库中的功能。

为什么是Python火了?

- 编程已经是我们生活中的重要组成元素
- 程序开发产生软件, 那什么是软件?
 - 软件 = 程序 + 数据 + 文档



- 程序员都用Python做什么？



- 计算机是根据指令操作数据的设备，具备**功能性**和**可编程性**两个基本特性：
 - 功能性指对数据的操作
 - 可编程性指可预测地、自动地执行一组指令
- 编程语言是计算机能够理解和识别用户操作意图的一种交互体系
 - 按照特定规则组织计算机指令，使计算机能够自动进行各种运算处理
- 低级编程语言：机器语言和汇编语言。
- 高级编程语言：更接近自然语言的程序设计语言。

- Python: 解释型语言 动态语言
- C++ : 编译型语言 静态语言

- **编译**是将源代码转换成目标代码的过程
 - 执行编译的计算机程序称为编译器
 - 编译是一次性地翻译，一旦程序被编译，不再需要编译程序或者源代码
- **解释**是将源代码逐条转换成目标代码同时逐条运行目标代码的过程
 - 执行解释的计算机程序称为解释器
 - 解释则在每次程序运行时都需要解释器和源代码
 - 解释执行需要保留源代码，程序维护方便

- Python: 解释型语言
- C++ : 编译型语言

动态语言
静态语言

- 动态类型语言
 - 数据类型的检查是在运行时做
 - 用动态类型语言编程时, 不用给变量指定数据类型, 该语言会在你第一次赋值给变量时, 在内部记录数据类型
- 静态类型语言
 - 数据类型的检查是在运行前 (如编译阶段) 做

- Python属于动态类型语言，解释器会根据赋值或运算来自动推断变量类型，且变量类型可随时变化
- 在Python中，**不需事先声明变量名及其类型**，直接赋值即可创建各种类型的对象变量

例如语句

```
>>>x = 3
```

创建一个整型变量x

创建了整型变量x，并赋值为3

```
>>>x = 'Hello World!'
```

新的赋值，不再是原来的x

创建了字符串变量x，并赋值为Hello World!

Python基础知识



- Python是一门跨平台、开源、免费的解释型高级动态编程语言
- Python支持
 - 函数式编程
 - 面向过程编程
 - 面向对象编程
- Python拥有几乎支持所有领域应用开发的成熟扩展库，可作为胶水语言

 C++	 JavaScript
 Java/C#	 PHP(Without MySQL)
 Ruby	 Pascal
 Perl	 Lisp
 Visual Basic	 Haskell
 Python	 C

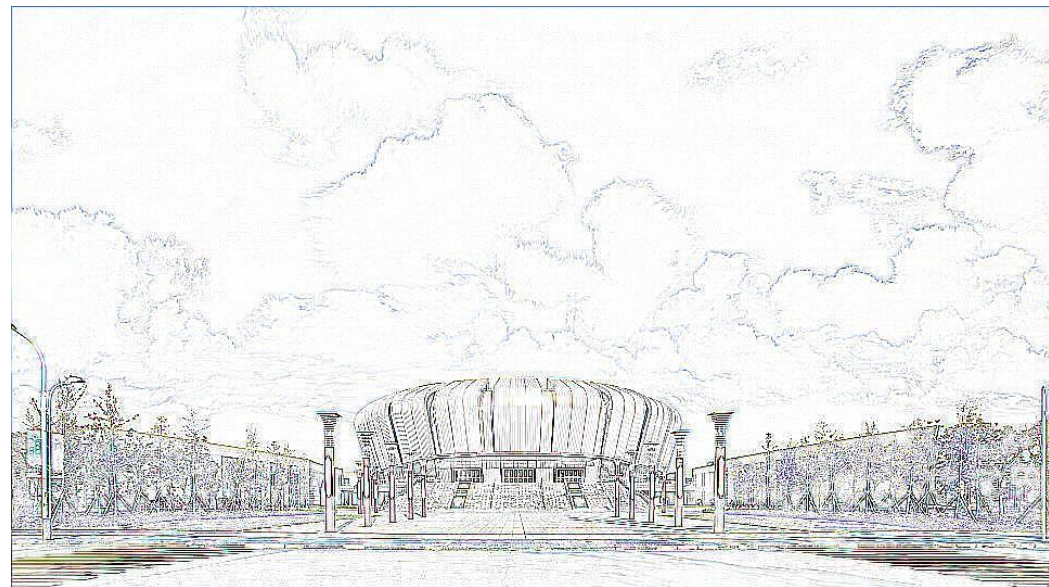
Python基础知识



Pillow库提供的素描滤镜



```
#coding=utf-8  
from PIL import Image, ImageFilter  
fig = Image.open('gym.jpg')  
sketch = fig.filter(ImageFilter.CONTOUR)  
sketch.save('sketch.jpg')
```



- `python -m freegames.snake`



贪食蛇游戏

- `python -m freegames.flappy`

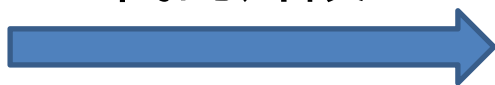


flappy bird游戏

- 贪食蛇游戏仅用45行有效代码!
 - 几行代码如何实现这些功能?



代码片段



```
from turtle import *
from random import randrange
from freegames import square, vector
food = vector(0, 0)
snake = [vector(10, 0)]
aim = vector(0, -10)
def change(x, y):
    "Change snake direction."
    aim.x = x
    aim.y = y
def inside(head):
    "Return True if head inside boundaries."
    return -200 < head.x < 190 and -200 < head.y < 190
def move():
    "Move snake forward one segment."
    head = snake[-1].copy()
    head.move(aim)
```



- 优势
 - 代码简洁，强制缩进提高可读性
 - 脚本语言，语句逐行执行
 - 跨平台 + 开源
 - 函数式编程 + 面向过程 + 面向对象
 - 易于学习，易于推广



- 产业应用-国际

- 谷歌系

- Google App Engine
 - code.google.com
 - Google earth
 - Google crawler
 - Google广告
 - YouTube

- Dropbox

- Reddit

- Instagram

- ...

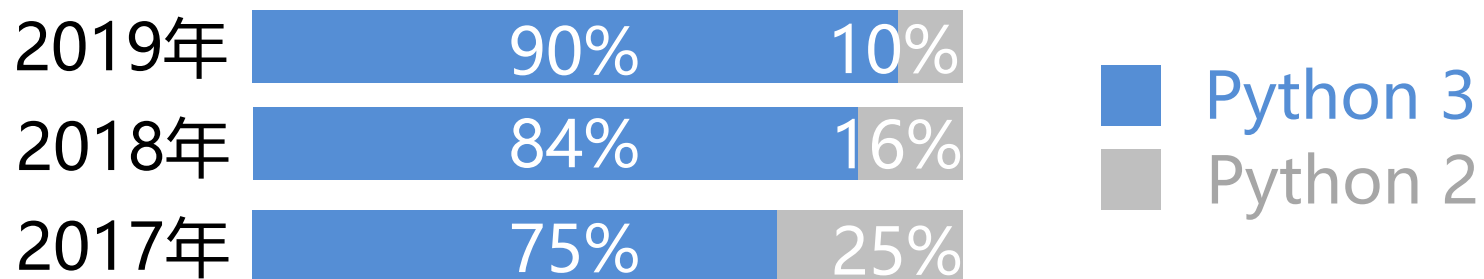
- 产业应用-国内

- 豆瓣

- 知乎

- 企业大量使用Python完成工作任务

- Python版本选择



- Python开发环境选择

- PyCharm
- VS Code
- Vim
- Jupyter Notebook
- Sublime text
- ...

Python基础知识



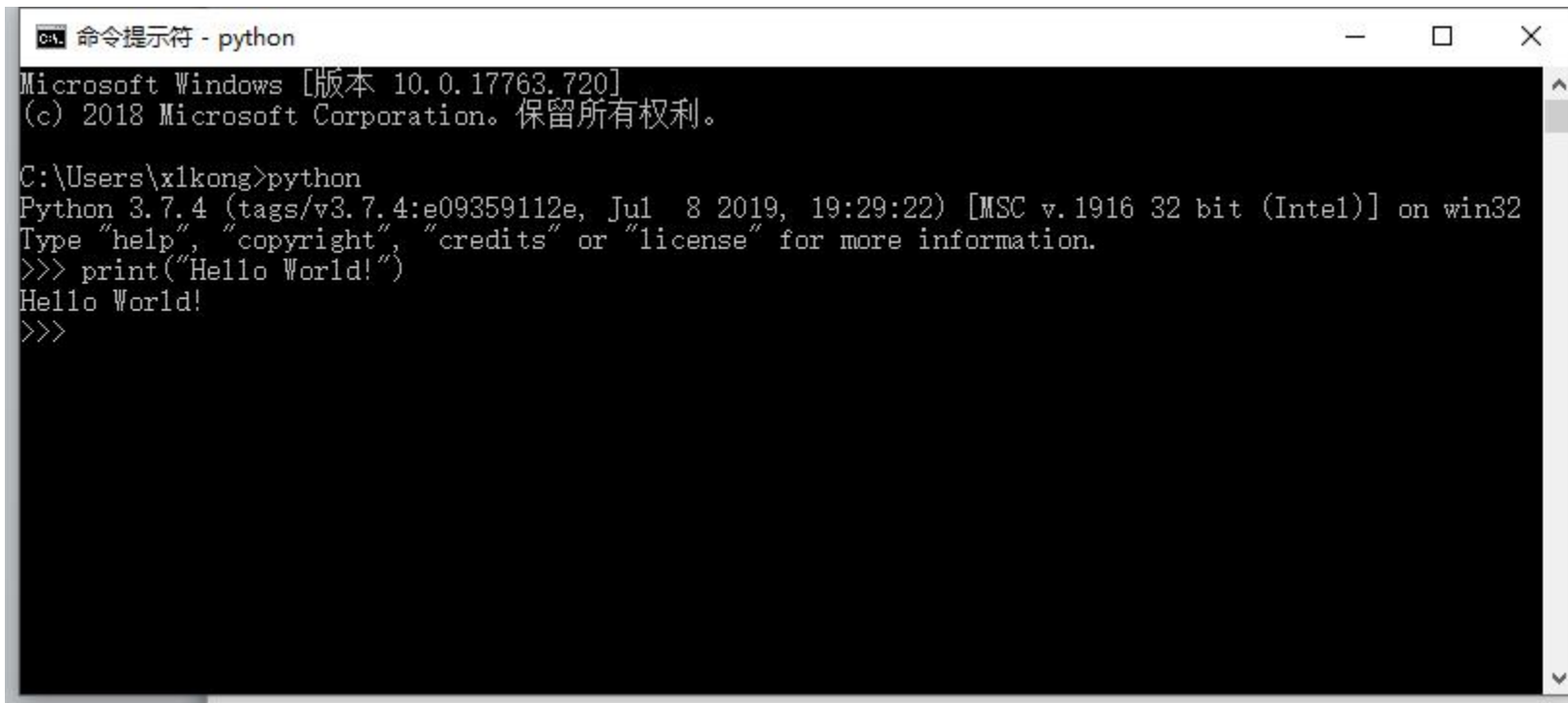
- Windows环境下安装
 - 在Python官网主页下载并安装Python基本开发和运行环境
<https://www.python.org/downloads/>



Python基础知识



- 使用方式一：Windows命令行
 - 键入python，启动交互式开发环境
 - 逐行键入程序

A screenshot of a Windows command prompt window titled "命令提示符 - python". The window shows the execution of the Python interpreter. The text displayed is as follows:

```
Microsoft Windows [版本 10.0.17763.720]  
(c) 2018 Microsoft Corporation。保留所有权利。  
  
C:\Users\xlkong>python  
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> print("Hello World!")  
Hello World!  
>>>
```

Python基础知识



- 使用方式二：运行代码文件
 - 在命令行运行python xxx.py

A screenshot of a Notepad window titled 'hellp.py - 写字板'. The window contains a single line of Python code: `print("Hello World!")`. The code is written in a monospaced font, and the cursor is positioned at the end of the line.

```
hellp.py - 写字板
文件 主页 查看
1 2 3 4 5 6
print("Hello World!")
```

A screenshot of a Windows Command Prompt window titled '命令提示符'. The window shows the output of running the Python script. The text displayed is: `Microsoft Windows [版本 10.0.17763.720]`, `(c) 2018 Microsoft Corporation. 保留所有权利。`, `C:\Users\xlkong>python hellp.py`, `Hello World!`, and `C:\Users\xlkong>`.

```
命令提示符
Microsoft Windows [版本 10.0.17763.720]
(c) 2018 Microsoft Corporation. 保留所有权利。

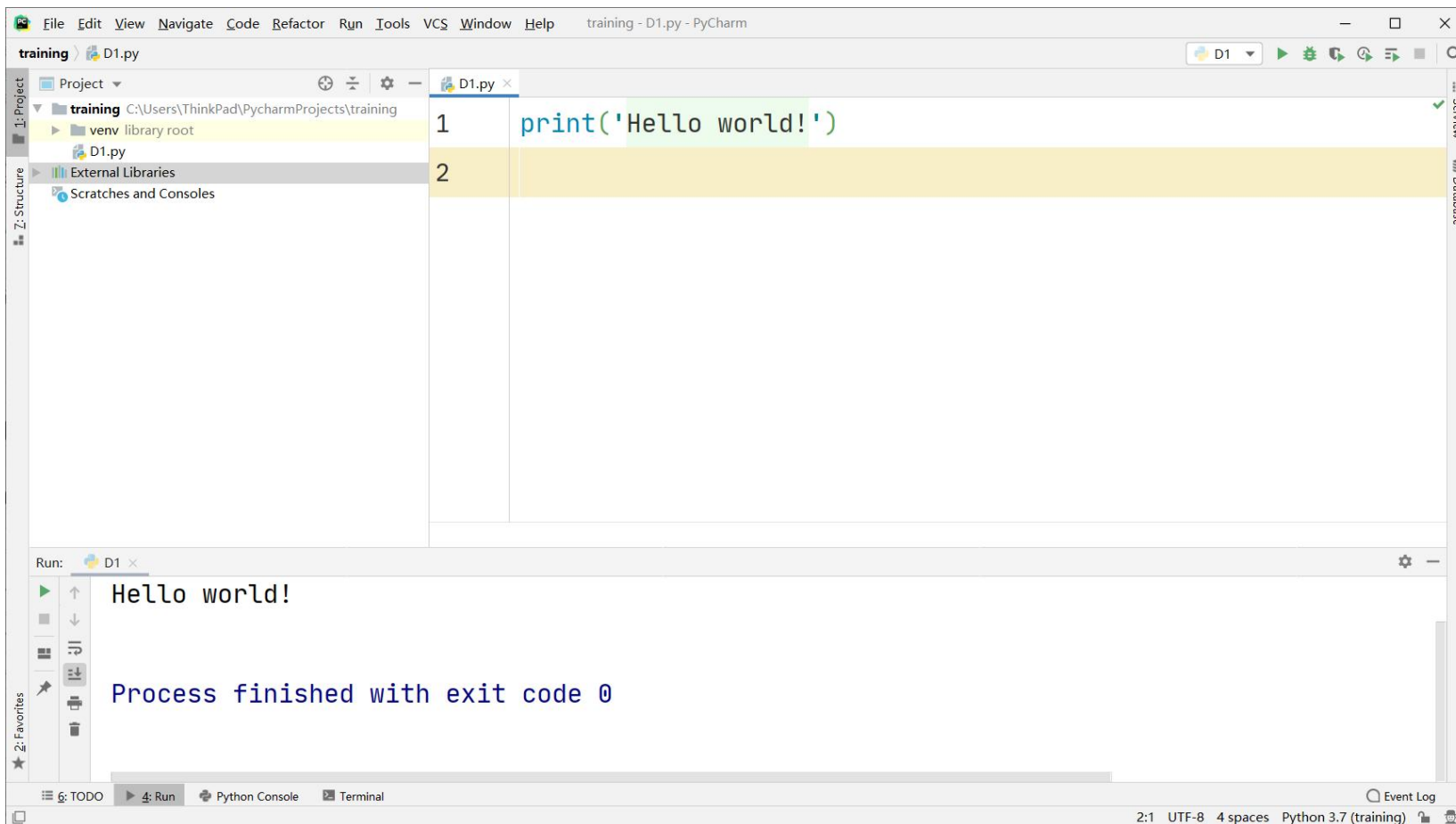
C:\Users\xlkong>python hellp.py
Hello World!

C:\Users\xlkong>
```

Python基础知识



- 使用方式三：集成开发环境 - PyCharm
 - 直接键入程序代码

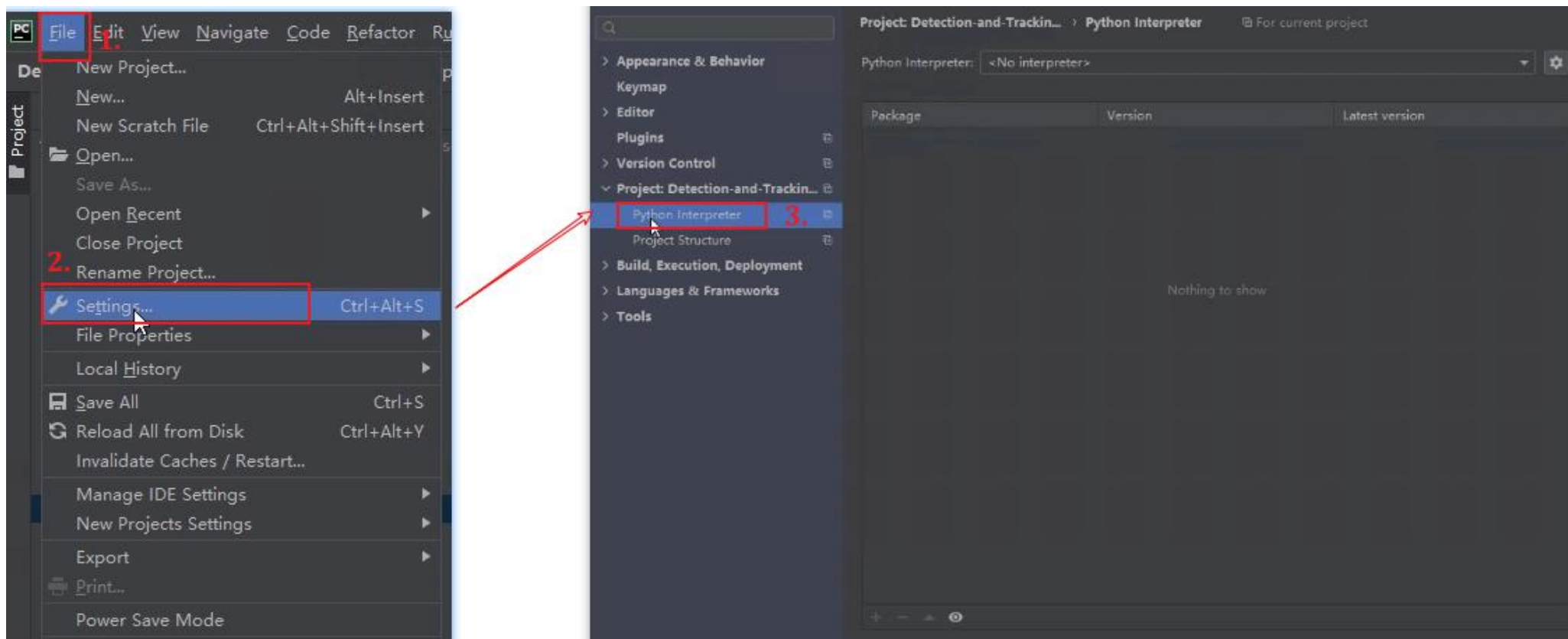


- Windows环境下安装
 - 在Python官网主页下载并安装Python基本开发和运行环境
<https://www.python.org/downloads/windows/>
 - 安装Anaconda包、环境管理器 (**推荐**)
<https://www.anaconda.com/download>
 - 安装PyCharm集成开发环境(IDE)
<https://www.jetbrains.com/pycharm/download/?section=windows>



- PyCharm中配置使用Anaconda环境

开始配置环境。点击“File”菜单，选择“Setting”选项，在“Project”中找到“Python Interpreter”，输入python的安装目录



- 搜索 “PyCharm中配置使用Anaconda环境” , 以下仅供参考
安装Anaconda手动配置anaconda环境变量 (path) 方法
<https://blog.csdn.net/fangweijiex/article/details/115825000>

在PyCharm中配置使用Anaconda环境

<https://blog.csdn.net/TuckX/article/details/115681862>

编程语言的基本要素



- **数据类型**：描述数据在内存存储中占据的空间大小，及其能够表示的范围
- **常量**：值不变的量
- **变量**：值可以改变的量
- **运算符**：连接数据的纽带
- **表达式**：由常量、变量、运算符和数据组成
- **标识符**：常量、变量、类等对象的命名
- **关键字**：具有特定含义的标识符
- **数据结构**：管理数据的方式
- **控制结构**：顺序、选择和循环等结构
- **函数**：处理数据的代码
- **输入/输出流**：基于输入输出语句，或函数实现的输入输出功能
- **解释器/编译器**：将代码转译成机器语言的工具

- IPO程序编写方法

- 输入数据

- 输入 (Input) 是一个程序的开始。程序要处理的数据有多种来源, 形成了多种输入方式, 包括: 文件输入、控制台输入、内部参数输入等等。

- 处理数据

- 处理 (Process) 是程序对输入数据进行计算产生输出结果的过程。计算问题的处理方法统称为“算法”。

- 输出数据

- 输出 (Output) 是程序展示运算成果的方式。程序的输出方式包括: 控制台输出、图形输出、文件输出等等。

- Python的对象模型
 - 对象是Python语言中最基本的概念
 - 许多内置对象可直接使用
 - 如数字、字符串、列表等
 - 非内置对象需要导入模块才能使用
 - 如正弦函数`sin()`，随机数产生函数`random()`等
- 输入数据、处理数据和输出数据时都需要以对象为载体。

- Python中的常用对象

对象类型	类型名称	示例	简要说明
数字	int, float, complex	1234, 3.14, 3+4j	数字大小无限制，内置支持复数运算
字符串	str	'Python', "Python"	使用单引号、双引号和三引号作定界符
列表	list	[1, 2, 3], ['a', [1,2]]	所有元素放在一对方括号中，其中元素可以是任意类型
字典	dict	{1:'sun', 2:'moon'}	所有元素放在一对大括号中，形式是 键: 值
元组	tuple	(1, 2, 3), (1,)	所有元素放在一对圆括号中，以逗号相隔，元素不可变
集合	set, frozenset	{1, 2, 3}	所有元素放在一对大括号中，以逗号相隔，元素不重复

- Python中的常用对象

对象类型	类型名称	示例	简要说明
布尔型	bool	True, False	逻辑值
空类型	Nonetype	None	空值
异常	Exception, ValueError...		Python内置大量异常类
文件		f=open('a.txt', 'rb')	open是内置函数中的一种
其他可迭代对象		map对象、filter对象...	具有惰性求值的特点
编程单元		函数、模块、类	类和函数都属于可调用对象，模块用来存放函数，类等

- 函数： 内置函数

<u>abs()</u>	<u>delattr()</u>	<u>hash()</u>	<u>memoryview()</u>	<u>set()</u>
<u>all()</u>	<u>dict()</u>	<u>help()</u>	<u>min()</u>	<u>setattr()</u>
<u>any()</u>	<u>dir()</u>	<u>hex()</u>	<u>next()</u>	<u>slice()</u>
<u>ascii()</u>	<u>divmod()</u>	<u>id()</u>	<u>object()</u>	<u>sorted()</u>
<u>bin()</u>	<u>enumerate()</u>	<u>input()</u>	<u>oct()</u>	<u>staticmethod()</u>
<u>bool()</u>	<u>eval()</u>	<u>int()</u>	<u>open()</u>	<u>str()</u>
<u>breakpoint()</u>	<u>exec()</u>	<u>isinstance()</u>	<u>ord()</u>	<u>sum()</u>
<u>bytearray()</u>	<u>filter()</u>	<u>issubclass()</u>	<u>pow()</u>	<u>super()</u>
<u>bytes()</u>	<u>float()</u>	<u>iter()</u>	<u>print()</u>	<u>tuple()</u>
<u>callable()</u>	<u>format()</u>	<u>len()</u>	<u>property()</u>	<u>type()</u>
<u>chr()</u>	<u>frozenset()</u>	<u>list()</u>	<u>range()</u>	<u>vars()</u>
<u>classmethod()</u>	<u>getattr()</u>	<u>locals()</u>	<u>repr()</u>	<u>zip()</u>
<u>compile()</u>	<u>globals()</u>	<u>map()</u>	<u>reversed()</u>	<u>__import__()</u>
<u>complex()</u>	<u>hasattr()</u>	<u>max()</u>	<u>round()</u>	

- 数字：常用运算

运算	结果
$x + y$	x 和 y 的和
$x - y$	x 和 y 的差
$x * y$	x 和 y 的乘积
x / y	x 和 y 的商
$x // y$	x 和 y 的商取整
$x \% y$	x / y 的余数
$-x$	x 取反
$+x$	x 不变
<code>abs(x)</code>	x 的绝对值或大小
<code>int(x)</code>	将 x 转换为整数
<code>float(x)</code>	将 x 转换为浮点数
<code>complex(re, im)</code>	一个带有实部 re 和虚部 im 的复数。 im 默认为0。
<code>c.conjugate()</code>	复数 c 的共轭
<code>divmod(x, y)</code>	$(x // y, x \% y)$
<code>pow(x, y)</code>	x 的 y 次幂
$x ** y$	x 的 y 次幂

- Python变量的类型

```
x = 3
```

```
print("变量赋值3, 其类型为", type(x))
```

```
x = "Hello World!"
```

```
print("变量又赋值Hello World!, 其类型为", type(x))
```

```
x = [1, 2, 3]
```

```
print("变量又又赋值[1, 2, 3], 其类型为", type(x))
```



变量赋值3, 其类型为 <class 'int'>

变量又赋值Hello World!, 其类型为 <class 'str'>

变量又又赋值[1, 2, 3], 其类型为 <class 'list'>

- 返回对象的内存地址（id 函数）
 - id(object)函数是返回对象object在其生命周期内位于内存中的地址
 - 为了提高内存利用效率，对于一些简单的对象，如int、float、字符串对象等，python采取重用对象内存的办法

```
c = "abc"
```

```
d = "abc"
```

```
print(id(c),id(d),id("abc"))
```

```
print('c == d:',c==d)
```

```
print('c is d:',c is d)
```

```
c == d: True
```

```
c is d: True
```

- is与==的使用区别

- is 比较的是两个实例对象是不是完全相同，它们是不是同一个对象，占用的内存地址是否相同
- == 是比较两个对象的内容是否相等，即两个对象的值是否相等，不管两者在内存中的引用地址是否一样

```
val2 = 2001
```

```
val3 = 2000 + 1
```

```
print(id(val3)==id(val2),val3==val2)      True True
```

```
val1 = 2000
```

```
val2 = 2001
```

```
val3 = val1 + 1
```

```
print(id(val3)==id(val2),val3==val2)      False True
```

- is与==的使用区别

```
a = [1, 2, 3]
b = [1, 2, 3]
print(id(a),id(b))
print('a == b:', a==b)
print("a is b", a is b)
```

```
a == b: True
a is b False
```

- 在Python中，允许多个变量指向同一个值，当修改其中一个变量的值以后，其内存地址将会变化，但这并不影响另一个变量。

```
>>> x = 3
>>> id(x)
265446608
>>> y = x
>>> id(y)
265446608
```

当x的值发生变化时

y的值不变、地址不变

```
>>> x = 3
>>> id(x)
265446608
>>> y = x
>>> id(y)
265446608
>>> x += 6
>>> id(x)
265446704
>>> print(y)
3
>>> id(y)
265446608
```

- Python采用基于值的内存管理方式，如果为不同变量赋予相同值，这个值在内存中只有一份，多个变量指向同一块内存地址

```
>>> x = 3
>>> id(x)
265446608
>>> y = 3
>>> id(y)
265446608
>>> x = [2, 3, 3, 3]
>>> id(x[1])
265446608
>>> id(x[1]) == id(x[2])
True
```

- 赋值语句
 - 首先把等号右侧表达式的值计算出来
 - 然后在内存中寻找一个位置把值存进去
 - 最后创建变量并指向这个内存地址
- Python中的变量不直接存储值，而是存储了值的内存地址或者引用，这也是变量类型可以随时改变的原因
- Python具有自动内存管理功能，对于没有任何变量指向的值，会自动删除
- 尽管如此，使用del命令删除不需要的值，关闭不再需要访问的资源，仍是一个好习惯



- 定义变量名时，需要注意如下问题：
 - 变量名须以字母或下划线开头，以下划线开头的变量有特殊含义
 - 变量名中不能含有空格以及标点符号
 - 不能使用关键字作变量名
 - 不建议使用系统内置模块名、类型名等作变量名
 - 变量名对英文字母的大小写敏感

Python基础知识-小结



- 程序设计语言基础知识
 - 低级语言 VS 高级语言
 - 解释型语言 VS 编译型语言
 - 动态语言 VS 静态语言
- Python的优势
 - 简洁、可读性强、跨平台、扩展库丰富、设计理念灵活
- Python的劣势
 - 运行速度慢、安全性差、多线程支持差、工具人
- Python的安装与使用（对象模型与变量）

- 数值四舍五入
 - 将浮点数四舍五入到固定的小数位数
- 对整数保留到十位、百位和千位
 - 例如，111保留到十位是110
- 格式化输出
 - 把浮点数保留两位小数点输出
 - 把整数转为长度为N的字符串，整数不足N位在前补零。例如，把111转为5位字符串，输出 “00111”

- 精确的小数计算
 - $(2.2 + 2.1) == 4.3$ 结果是True还是False?
- 无穷大、NaN (not a number) 和epsilon (浮点相对精度)
 - 如何表示正无穷大、负无穷大和NaN
 - 判断某个数是否为无穷大或NaN
 - `float('inf') == float('inf')` 结果是True还是False?
 - `float('nan') == float('nan')` 结果是True还是False?
 - 无穷大/NaN和数字的运算
 - 如何获取epsilon数值

- 复数数值运算
 - 用两种方法初始化表示复数的变量
 - 获取复数的实部、虚部和共轭
 - 实现复数的四则运算和求模
 - 实现基于复数的复杂运算，比如三角函数、指数函数
- 用分数计算
 - 定义分数变量
 - 获取分数的分母、分子
 - 分数和小数互相转化