

PYTHON Advanced Programming

何志权

1

2

人工智能时代的利器



3

Why Python



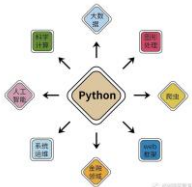
Python的设计哲学:



Python开发者的哲学:

“用一种方法，最好是只有一种方法来做一件事”。

在设计Python语言时，如果面临多种选择，Python开发者会选择最优雅、最明确、最简洁的方法，而选择明确的方法或者很少有效的方法。



CC BY-SA 4.0

5

Get a Big Picture

信息时代的下一波浪潮

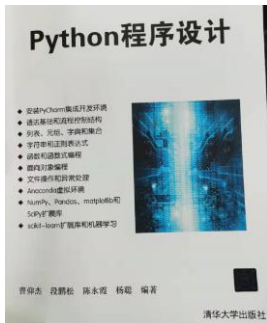
https://www.bilibili.com/video/BV1e64y1Y7d2?p=1&share_medium=android&share_plat=android&share_session_id=01139a49-d7d-4cd1-973b-g19b533ce530&share_source=WEIXIN&share_tag=s_i×tamp=1630671067&unique_k=mEm82f

Why Python

- 完全免费，绝大多数科学计算相关扩展库也都是免费的，
- 大多数代码是开源的
- 良好的生态：Matlab不如Python。比如3D的绘图工具包，比如GUI，比如更方便的并行，使用GPU，Functional等等。长期来看，Python的科学计算生态会比Matlab好
- 语言更加优美。如果有一定的OOP需求，构建较大一点点的科学计算系统，直接用Python比用Matlab混合的方案肯定要简洁不少。
- python作为一种通用编程语言，可以做Web，搞个爬虫，编个脚本，写个小工具用途很广泛
- 深度学习的广泛应用，让python称为市场宠儿。矩阵操作

4

课程教材



6

课程内容和目标

- 课程内容
 - Python 语法基础和编程
 - Numpy 数值计算
 - Python 可视化编程
- 课程目标
 - 熟悉 Python 语言，能熟练编程（最低要求）
 - 提高工程思维能力、动手能力
 - 应用Python 解决实际问题的能力

7

学习方式与课程考核

- 学习方式：重在练习
- 课堂授课，讲解
 - 课后作业
 - 基本实验 + 综合实验
 - 课程设计

班内查重!!!

8

交流互助

- Mail: zhiquan@szu.edu.cn

- QQ 群



群名称: PythonAD2021
群 号: 202536396

9

Before stepping into

Prepare for AI: AI = ? , AI Engineer

什么是工程师思维、工程师文化？

实现业务的自动化。DON'T REPEAT YOURSELF !

编码只是一种工具

一种系统化、结构化的思维。

最小化的编码解决更大范围的问题

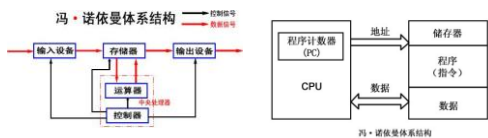
10

Before stepping into

站在计算机的角度去思维、掌握计算机沟通的语言

自然语言、数学、编程语言

Computer Inside



11

Before stepping into

- 数据搬移：内存中的数据操作
 - 赋值操作: a = 3
 - 支持的数据类型

- 数学、关系运算：
 - 加减乘除，比较，etc

- 运行逻辑
 - 条件语句，循环语句

- 一门语言的核心
- Building block
- 算法：有效的组织核心元素

12

进入Python编程世界

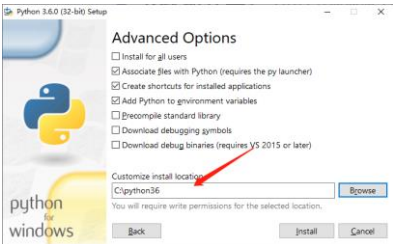
- 下载和安装: <http://www.python.org>



13

进入Python编程世界

- 下载和安装: <http://www.python.org>



14

进入Python编程世界

- 下载和安装: <http://www.python.org>

- Python 2.x vs Python 3.x
 - 编码方面: Python3.x默认utf-8。
 - 语法方面有改动, 数据类型方面有调整。
 - 面向对象、异常处理和模块等方面也有改动。
 - 本教学使用Python3.6

15

进入Python编程世界

- 安装opencv:
pip install opencv-python
- 安装matplotlib:
pip install matplotlib
- 在Python 命令行里检查安装完好

16

Python 使用

- 命令行方式
- 脚本式编程
 - 命令行运行脚本文件: python test.py
 - IDE内部运行脚本

17

20210913

18

Python 语法基础

- Python的代码缩进：通过代码缩进来保持对应语句逻辑的一致性，不一致的缩进会导致代码运行的错误

```
1 if a > b:
2     a = b
3 else:
4     b = a
```

```
1 if a > b:
2     a = b
3 else: # 此处缩进量错误，导致代码运行出错
4     b = a
```

- Python的注释：
 - 单行注释采用#号表示，
 - 多行注释使用''' XXXX '''表示，前后各三个引号

```
1 '''
2 该程序的功能是：
3 简单的说明多行注释方法而已
4 '''
5 if a > b:
6     a = b
7 else:
8     b = a
```

19

Python 语法基础

多行语句

可以使用斜杠（\）将一行的语句分为多行显示，如下所示：

```
a = 1 + 2 + 3 + 4 + 5 + 6 + 7 \
    + 8 + 9 + 10
print(a)
```

语句中包含[], {} 或 () 括号就不需要使用多行连接符。如下实例：

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

20

Python基本数据类型

Python有五个标准的数据类型：

- 不可变的
- Numbers（数字）：int, bool, float, complex
 - String（字符串）
 - Tuple（元组）

- 可变的
- List（列表）
 - Dictionary（字典）
 - Set（集合）

type(): 检查变量的类型
isinstance(a, int)

21

Python 语法基础

变量命名规则与关键字

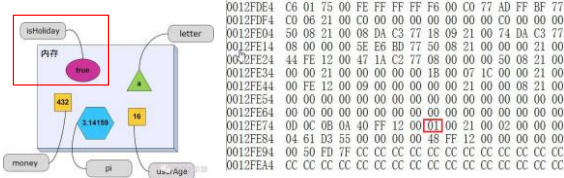
- 变量名的长度不受限制，其中的字符必须是字母、数字或下划线（_），不能使用空格、连字符、标点符号、引号或其他字符
- 变量名的第一个字符不能是数字，必须是字母或下划线。
- Python区分大小写，。
- 不能将Python关键字用作变量名。

and	as	assert	break	class
continue	def	del	elif	else
except	exec	False	finally	for
from	global	if	import	in
is	lambda	not	None	or
pass	print	raise	return	try
True	while	with	yield	

22

Python 基本数据操作

变量：指向一个内存地址块



编码 — 编译 — 链接 — 优化

编译：自然语言 → 机器指令，为变量、数据分配好内存地址

23

Python 基本数据操作

赋值操作：

```
a = 1          # int
b = 1.2        # float

c = 1 + 2j     # 只能用j, 或 J.
# c.real, c.imag
d = 'this is a string'
```

```
>>> x=5
>>> x
5
>>> y='cat'
>>> y
'cat'
>>> x=y
>>> x
'cat'
>>> y
'cat'
```

多元赋值：a = b = c = 10 or a, b, c = 1, 2, "AI"

24

算术运算符

Python中的基本算数运算与C、C++、C#、Java等语言基本一样，只有一个例外：2**3表示的是2的三次方，此处**表示次方运算

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

三种类型存在一种逐渐“扩展”的关系：**整数 -> 浮点数 -> 复数**（整数是浮点数特例，浮点数是复数特例）

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

内置的数值函数：abs, fabs, ceil, floor, exp, log, log10, round, sqrt etc.

25

赋值运算符

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

a = 21
b = 10
b += a

26

比较运算符

返回True, False

关系运算符	关系表达式	描述	优先级
<	x<y	小于	优先级相等；但优先级大于==和!=
>	x>y	大于	
<=	x<=y	小于等于	
>=	x>=y	大于等于	优先级相等
==	x==y	等于	
!=	x!=y	不等于	

a = 21
b = 10
c = (a == b)

27

逻辑运算符

True or False: bool(2), bool(0), bool(-2)

and	x and y	布尔“与”- 如果 x 为 False, x and y 返回 x 的值, 否则返回 y 的计算值。
or	x or y	布尔“或”- 如果 x 是 True, 它返回 x 的值, 否则它返回 y 的计算值。
not	not x	布尔“非”- 如果 x 为 True, 返回 False, 如果 x 为 False, 它返回 True。

```
print(1 and 2) # 2
print(3 and 0) # 0
print(0 and 2) # 0
print(3 and 2) # 2
print(0 and 0) # 0
```

```
print(1 or 2) # 1
print(3 or 2) # 3
print(0 or 2) # 2
print(0 or 100) # 100
print(0 or 0) # 0
```

28

位运算符

运算符	描述	实例
&	按位与运算符：参与运算的两个值,如果两个相应位都为1,则该位的结果为1,否则为0	(a & b) 输出结果 12, 二进制解释: 0000 1100
	按位或运算符：只要对应的二个二进位有一个为1时,结果位就为1。	(a b) 输出结果 61, 二进制解释: 0011 1101
^	按位异或运算符：当对应的二进位相异时,结果为1	(a ^ b) 输出结果 49, 二进制解释: 0011 0001
~	按位取反运算符：对数据的每个二进位取反,即将1变为0,0变为1。~x 类似于 -x-1	(~a) 输出结果 -61, 二进制解释: 1100 0011, 在一个有符号二进制数的补码形式。
<<	左移动运算符：运算数的各二进位全部左移若干位,由“<<”右边的数指定移动的位数,高位丢弃,低位补0。	a << 2 输出结果 240, 二进制解释: 1111 0000
>>	右移动运算符：把“>>”左边的运算数的各二进位全部右移若干位,“>>”右边的数指定移动的位数	a >> 2 输出结果 15, 二进制解释: 0000 1111

29

位运算符

位运算：a & b, a | b, a ^ b, ~a, a >> 2, a << 2

a 为 60, b 为 13

bin(a)
oct(a)
hex(a)

a2 = 0b00111100

a = 0011 1100
b = 0000 1101

a&b = 0000 1100
a|b = 0011 1101
a^b = 0011 0001
~a = 1100 0011

30

成员运算符

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中；如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x 不在 y 序列中；如果 x 不在 y 序列中返回 True。

```
tmpList = [1, 2, 3, 4, 5 ]
a = 5
b = (a in tmpList)
```

身份运算符

身份运算符： is , is not
is 是判断两个标识符是不是引用自一个对象
xx = A is B

```
a = 3
b = 3
print(a is b)

b = 4
print(a is b)
```

is 与 == 区别：
is 用于判断两个变量引用对象是否为同一个
， == 用于判断引用变量的值是否相等。

Python 字符串

字符串可以使用单引号'，双引号"，三个单引号'''，三个双引号"""表示；
例如：'abc' "abc" """abc"""

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"

```
>>> greet="Hello John"
>>> print(greet[2])
1
>>> x=8
>>> print(greet[x-2])
J
>>>
>>> greet[0:3]
'Hel'
>>>
```

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

-4 -3 -2 -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[-4:-1]: -4, -3, -2
greet[-4:]: -4 到末尾的全部
greet[a:b:c]
```

Python 字符串

```
>>> "pine" + "apple"
'pineapple'
>>> 3 * "pine"
'pinepinepine'
>>>
```

len()函数能否返回一个字符串的长度

```
>>> len("pine")
4
>>> len("祖国，您好！")
6
```

大多数数据类型都可以通过str()函数转换为字符串

```
>>> str(123)
'123'
>>> str(123.456)
'123.456'
>>> str(123e+10)
'1230000000000.0'
```

Python 字符串

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()	通过指定字符连接序列中元素后生成的新字符串。
<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

Python 字符串

字符串格式化

Python2
print("%.2f, %.2f, %.2f" %(a,b,a))

Python3
print("{0:.2f}, {1:.2f}, {0:.2f}".format(a,b))
#冒号前面的数字代表取第几个变量

```
print("{0}, {1}, {0}".format(a,b))

print("{:.2f}, {:.2f}, {:.2f}".format(a,b,a))
print("", "", "").format(a,b,a))
```

Practice

```
a = 10, b = 20, 计算 a + b, b / a, b // a, a ** 2
1. c = (a > b)
2. d = (a != b)
3. e = ((a > b) and (a != b))
4. f = ((a > b) or (a != b))

a = "Hello"
b = "Python"
a[2] = ?      a[1:4] = ?
a[-4:-1] = ?  a[-4:] = ?
a+b = ?      a*3
d = a.replace('el', 'xxx')    d = ?
```

37

Tuple 操作

Tuple不能改写，只能访问

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7)
print(tup1[0]) # tup1[-3:-1], tup1[1:3], tup1[2:]
成员函数只有count和index 两个函数, dir(tup) 查看
```

内置元组函数

- 1、cmp(tuple1, tuple2): 比较两个元组元素。
- 2、len(tuple): 计算元组元素个数。
- 3、max(tuple): 返回元组中元素最大值。
- 4、min(tuple): 返回元组中元素最小值。
- 5、tuple(seq): 将列表转换为元组。

S[i:j:k]: 从索引为直
到索引为j-1, 每隔k
个元素索引一次

39

Python数据类型转换

```
type conversion:
str()
int()
float()
list()
tuple()

查看数据类型:
• type(x),
• x.__class__
```

41

Python tuple and list

- list是python内置的有序的列表，可以随时添加和删除其中的元素
- tuple和list的最大的区别就是 tuple一旦被创建 就无法修改
- 以逗号隔开

```
tup = (1,)      # 一个元素的tuple
tup = (1,2, ["a","b","c"], "a")

lst = [1,2, ["a","b","c"], "a"]
lst = []
```

38

List 操作

```
list = ['physics', 'chemistry', 1997, 2000]
print(list[0]) # list[1:5], list[2:], list[-4:-1]

list.append('Google') # 末尾添加元素
del(list[2])          # 删除第三个元素
list.insert(index, obj) # 将对象插入到指定位置
list.extend(seq)      # 两个序列拼接, 和+类似
list.reverse()        # 反向排列
list.index(obj)       # 找出首次出现obj的下标
list.sort()           # 元素排序
```

1. cmp(list1, list2): 比较两个列表的元素
2. len(list): 列表元素个数
3. max(list): 返回列表元素最大值
4. min(list): 返回列表元素最小值
5. list(seq): 将元组转换为列表

40

Python 流程控制

```
for i in range(1, 5):
    print(i)
else:
    print('The for loop is over')
```

```
while True:
    s = input('Enter something : ')
    if s == 'quit':
        break
    print('Length of the string is', len(s))
print('Done')
```

```
if 判断条件1:
    执行语句1.....
elif 判断条件2:
    执行语句2.....
elif 判断条件3:
    执行语句3.....
else:
    执行语句4.....
```

42

Python 流程控制

Demo & Practice

43

Python 函数

```
def functionname(name, age):  
    "函数_文档字符串"  
  
    函数体  
  
    return [expression]
```

```
def max(a, b):  
    if a > b:  
        return a  
    else:  
        return b  
  
a = 4  
b = 5  
print(max(a, b))
```

44

Python 函数参数

必备参数须以正确的顺序传入函数。调用时的数量必须和声明时的一样。

关键字参数和函数调用关系紧密，函数调用使用关键字参数来确定传入的参数值。使用关键字参数允许函数调用时参数的顺序与声明时不一致，因为Python 解释器能够用参数名匹配参数值。

```
def printinfo( name, age ):  
    "打印任何传入的字符串"  
    print ("名字: ", name)  
    print ("年龄: ", age)  
    return  
  
printinfo( age=50, name="runoob" )
```

45

Python 函数参数

默认参数, 调用函数时，默认参数的值如果没有传入，则被认为是默认值。

```
def printinfo( name, age = 35 ):  
    "打印任何传入的字符串"  
    print ("名字: ", name)  
    print ("年龄: ", age)  
    return  
  
printinfo( name="runoob" )
```

46

Python 函数

全局变量和局部变量

- 定义在函数内部的变量拥有一个局部作用域，定义在函数外的拥有全局作用域。
- 局部变量只能在其被声明的函数内部访问，而全局变量可以在整个程序范围内访问。

```
def myfun( arg1, arg2 ):  
    var1 = arg1 + arg2  
    var2 = arg1 - arg2  
    return (var1,var2);
```

47

Python IO

48

Python IO-文件操作

文本文件读写

```
fid = open('mydata.txt','r')
lines = fid.readlines()
fid.close()

fid = open('mydata.txt','w')
fid.write('%3d %3d\n' % (1,2))
fid.close()
```

49

数据结构

程序 = 数据结构 + 算法

数据结构：管理数据的方法，和语言无关

功能和目的：快速的定位、查找，牺牲空间复杂度，提升时间复杂度

51

数据结构

算法的复杂度

常数阶O(1)
i = 1;
++;
j++;
m = i + j;

线性阶O(n)
for item in aList:
 item += 1

对数阶O(logN)
int i = 1;
while(i<n) {
 i = i * 2;
}

平方阶O(n²)
for(x=1; i<=n; x++) {
 for(i=1; i<=n; i++) {
 j = i;
 j++;
 }
}

53

2021-09-27

Python 数据结构

数据结构

算法的复杂度

常见的时间复杂度量级有：

- 常数阶O(1)
- 对数阶O(logN)
- 线性阶O(n)
- 线性对数阶O(nlogN)
- K次方阶O(n^k)
- 指数阶(2^n)

从上至下依次的时间复杂度越来越大，执行的效率越来越低

50

数据结构

常见的数据结构包括：

线性表、栈、队列、串、数组、二叉树、树、图、查找表等

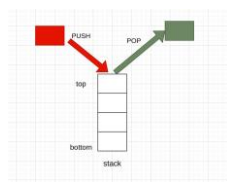
54

数据结构-list

```
list = ['physics', 'chemistry', 1997, 2000]

list.append('Google') # 末尾添加元素
del(list[2])          # 删除第三个元素
list.insert(index, obj) # 将对象插入到指定位置
list.index(obj)       # 找出首次出现obj的下标
list.pop()            #
list.extend(seq)      # 两个序列拼接，和+类似
list.reverse()        # 反向排列
list.sort()           # 排序
```

数据结构-stack

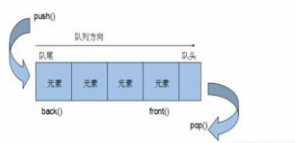


后进先出 (LIFO)

用list实现stack的操作函数: push(x), pop()

数据结构-队列

Queue



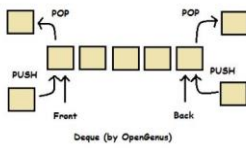
先进先出 (FIFO)

用list实现stack的操作函数: push(x), pop()

数据结构-双向队列

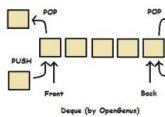
Deque

Deque是一种由队列结构扩展而来的双端队列(double-ended queue)，队列元素能够在队列两端添加或删除。因此它还被称为头尾连接列表(head-tail linked list)



数据结构-双向队列

Deque



用list实现deque的数据push, pop操作

数据结构-字典

- 字典用“{”标识，由索引(key)和它对应的值value组成
- 字典元素是通过键来存取

```
d = {'abc': 123, 98.6: 37}
```

```
d['abc'] or d.get('abc')
d['abc'] = 321
d.keys() : ['abc', 98.6]
d.values() : [123, 37]
d.items() : [('abc', 123), (98.6, 37)]
d.clear()

d.pop(key)
```

数据结构-字典

练习：合并两个字典

```
d1 = {'a':1, 'b':2}
d2 = {'c':3, 'd':4}
```

61

数据结构-集合

集合 Set: 数据不重复, 无序

```
set(['a', 'b', 'c', 'a', 1,2,1])
```

- 交集: $x \cap y$, 返回一个新的集合, 包括同时在集合 x 和 y 中的共同元素。
- 并集: $x \cup y$, 返回一个新的集合, 包括集合 x 和 y 中所有元素。
- 差集: $x - y$, 返回一个新的集合, 包括在集合 x 中但不在集合 y 中的元素。
- 补集: $x \Delta y$, 返回一个新的集合, 包括集合 x 和 y 的非共同元素。

62

数据结构-集合

练习:

```
x = set('eleven') y = set('twelve')
```

- $x \cap y$ #交集
- $x \cup y$ #并集
- $x - y$ #差集
- $y - x$ #差集
- $x \Delta y$ #补集
- $y \Delta x$ #补集

63

字典应用-词频分析案例

In March, I challenged American biotechnology companies to develop a rapid, inexpensive, and accurate test. Abbott Laboratories has now delivered. My Administration recently announced the purchase of 150 million Abbott Laboratories BinaxNOWTM rapid point-of-care tests. I am especially proud these tests are being Made in America, by Americans, and for Americans. Moreover, the swabs used for this test are available because of my Administration's use of the Defense Production Act (DPA) to invest in domestic manufacturing. This is a major development that will help save more lives by further protecting America's most vulnerable and allowing our country to get Americans back to work and children back to school.

- 找出每个单词出现的频率, 区分大小写, 标点符号忽略
- 找出出现频率最高的前5名的单词

64

字典应用-词频分析案例



65

Python 面向对象

66

