https://eagleslab.cowtransfer.com/s/c2d686fee3ac40

计算机的组成

CPU

- 计算机大脑，负责处理数据
- CPU只能处理二进制指令

内存

- 用来临时存放文件或指令的
- 内存是硬盘和CPU之间的缓存，CPU处理速度极快，硬盘速度跟不上
- 内存中的数据必须要带电保存，一旦内存断电了，数据全部丢失
- 内存中的数据往往是完全解压状态
- 内存中的代码或指令，一般称之为进程

硬盘

- 用来永久存放文件
- 硬盘中的软件或者代码，一般称之为程序

操作系统

- 用来衔接软件和硬件
- 不同的操作系统对软件有不同的标准，比如windows就是exe，Linux就是rpm或者deb

编程发展与应用

各种编程语言市场排名https://www.tiobe.com/tiobe-index/?tdsourcetag=s_pctim_aiomsg

python的分类

python2.x、python3.x的区别

- python2.x
    - 代码重复度高，规范混乱，较为复杂
- python3.x
    - 代码简洁，规范语法，比较简单
    - 与python2.x不兼容

编程语言的分类

按照程序运行的形式进行分类

`print("hello world")` --> `10101110110001110111`

- 编译型
    - 代表：c、c++、go
    - 在代码运行前，需要将所有的代码表达意思全部翻译为二进制，然后再执行
    - 优点：运行效率高
    - 缺点：开发难度大，不好维护，无法跨平台
- 解释型
    - 代表：python、php、javascript
    - 运行一行代码，翻译一行代码
    - 优点：开发简单，跨平台
    - 缺点：运行效率低，跨平台需要运行环境
- 混合型
    - 代表：java、c#
    - 既有全部翻译的行为，又有一行行解释运行的行为

python环境的安装

略，记得勾上最后一行的√

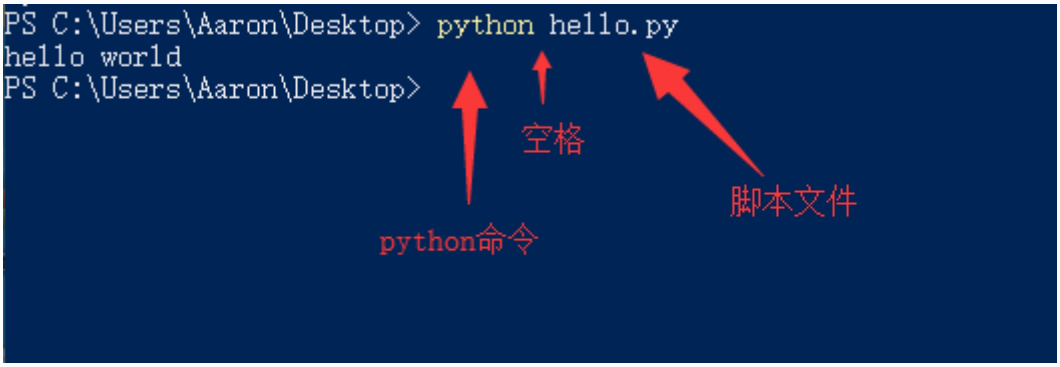
python程序的运行

- python代码全部记录在文本文档中，在桌面创建一个记事本，然后输入`print("hello world")`，引号必须是英文的字符
- 重命名这个记事本为`hello.py`，然后再桌面上打开`cmd`，运行`python hello.py`
- 建议电脑上安装`notepad++`或者`vscode`，这种高级记事本会帮助我们修改代码的时候不会出现字符错误

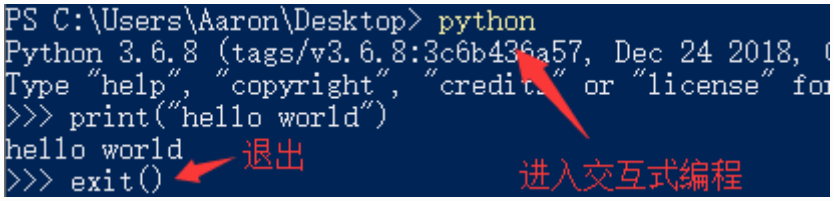
python解释器种类

- cpython
    - 用c语言实现将代码解释成二进制，从官网上下载就是cpython
- jpython
    - 将python代码解释成java的字节码

- ipython
    - 和cpython一致，不过允许交互编程
    - 编写一行代码就立即运行这行代码
- 脚本式编程
    - 用来编写大型程序

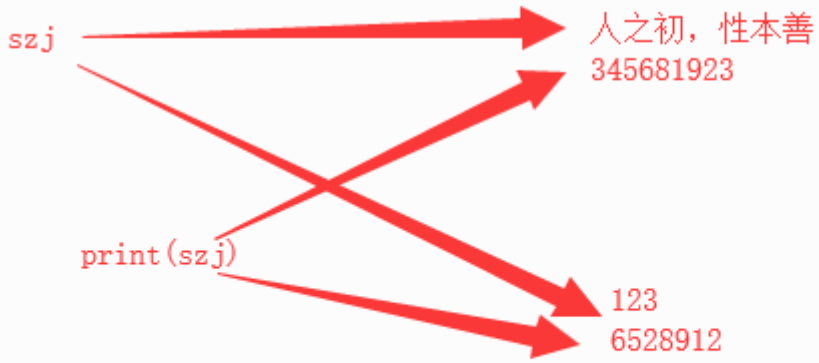


- 交互式编程



变量

将内存中的数据贴上一个标签，这个标签就是变量，在后续程序中可以直接对变量名进行操作。



- 变量名只能是 字母、数字、下划线任意组合组成(不推荐中文变量名)
- 变量名不能以数字开头
- 变量名不能使用关键字

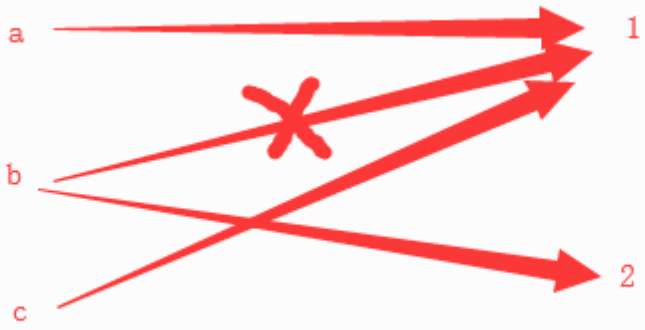
```
['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else',
'except', 'exec','finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'not', 'or', 'pass', 'print', 'raise','return', 'try', 'while', 'with',
'yield']
```

- 变量名在选取的时候需要有一定的意义
    - 推荐使用驼峰命名，下划线命名
        - `hp_of_cs` 下划线命名
        - `HpOfCs` 驼峰命名

```
a = 1
b = a
c = b
b = 2
print(a,b,c)
```

运行结果

```
1 2 1
```



常量

在python中，是没有绝对不变的常量的，但是为了对其他语言妥协，勉为其难的将全部大写的变量名称为常量。

```
import math
PI = math.pi
print(PI)
```

定义一个常量PI(表示圆周率)，运行结果

```
3.141592653589793
```

注释

- 增加代码可读性，标注代码的功能
- 在代码中添加注释，不会影响代码执行结果

```
# 单行注释
'''
可以使用三个单引号来
注释多行的内容
'''
```

程序输入输出

- 输出一般指的是在控制台显示内容
    - `print()`在()写上想输出的内容
- 输入就是让程序处于阻塞状态，等待接收用户键入的内容

```
name = input("请输入姓名:")
print(name)
```

数据类型基础

- 如果出现一个数据，人类是会自动将其分类，但是程序特别傻，必须要指定是什么类型

```
镇江市      地名
123     数字
True        单词表示真的
```

str(字符串)

- 在python中，所有字符串都是用单引号，双引号或三引号包起来的
- 对于python而言，字符串可以是任何字符，并且都没有意义
- 字符串是无法进行数学运算的

```
a = "123"
print(type(a),a)
```

```
a = "123"
b = "456"
print(a+b)
# python对字符串可以进行+操作，只是将其拼接一下
```

- 将其他数据类型转换为字符串

```
a = 1
b = str(a)
print(type(b))
```

int(整数)

在64位的python中，整数的位数定义的时候最大支持-2^{63}~2^{63}-1

```
a = 123
b = 321
print(a+b*a/b)
print(type(a))
```

- 只能将数字的字符串转换为整数

```
a = "123"
b = int(a)
print(type(b))
```

bool(布尔型)

- 布尔型数据只有两个
    - `True`
        - 不为空，就是`True`
    - `False`
        - 空或者0，都是`False`

```
a = True
print(a,type(a))
```

逻辑判断

- and、or、not
- 优先级为：not > and > or

and

- python中如果是`<条件1> and <条件2>`
    - 条件1为真，那么直接返回条件2的内容
    - 条件1为假，那么直接返回条件1的内容

```
print(1 and 2)      # 运行结果是2
```

or

- python中如果是`<条件1> or <条件2>`
    - 条件1为真，那么直接返回条件1的内容
    - 条件1为假，那么直接返回条件2的内容

not

- 对当前的结果取反

```
print(1 and 2 or 0 and not 3)
```

运算符

符号	算数
+	
-	
*	
/	
%	取余数
**	次方
//	整除结果
==	比较是否相等
!=	比较是否不相等
>	
<	
>=	大于等于
<=	小于等于

流程控制

if(判断)

- if主要用于逻辑判断，在python中，不同的条件可以让程序走不同的分支
- 如果匹配if的条件，那么执行的代码前面必须有4个空格，python对空格有严格的规定

```
if <条件>:
    执行的代码
```

```
name = input("输入姓名: ")
if name:
    print("欢迎"+name)
print("程序运行结束")
```

```
if <条件>:
    执行的代码
else:
    执行的代码(这边是不符合条件)
```

```
name = input("输入姓名: ")
if name:
    print("欢迎"+name)
else:
    print("姓名不能为空")
print("程序运行结束")
```

```
if <条件1>:
    执行的代码
elif <条件2>:
    执行的代码
elif <条件3>:
    执行的代码
..............
else:
    执行的代码(这边是不符合条件)
```

```
# 输入成绩
# 90-100 优秀
# 80-89 良好
# 60-79 及格
# 40-59 不及格
# 0-39 请家长

score = int(input("请输入成绩: "))
if 100 >= score >= 90:
    print("优秀")
elif 90 > score >= 80:
    print("良好")
elif 80 > score >= 60:
    print("及格")
elif 60 > score >= 40:
    print("不及格")
elif 40 > score >= 0:
    print("请家长")
else:print("请输入正确的成绩")
```

正常的写法

```
# 写一个登录程序
# 先验证验证码正确，再判断用户名是否正确
# 用户正确再判断密码是否正确
# 最后显示登录成功
# 比如用户名admin密码是admin888验证码是qwer

true_auth_num = 'qwer'
true_admin = 'admin'
true_password = 'admin888'

auth_num = input('请输入验证码')
```

```
if auth_num == true_auth_num:
    print('验证码正确')
else:
    print('验证码错误')
    exit()

auth_admin = input('请输入用户名')
if auth_admin == true_admin:
    print('用户名正确')
else:
    print('用户名错误')
    exit()
```

可以使用if嵌套

```
yzm = input("输入验证码: ")
if yzm == 'qwer':
    name = input("输入用户名: ")
    if name == "admin":
        password = input("输入密码: ")
        if password == "admin888":
            print("登录成功")
        else:print("密码错误")
    else:print("用户错误")
else:print("验证码错误")
```

while(循环)

- 当有代码需要重复的执行情况下，可以用到循环
- 当循环条件一直有效的情况下，会一直循环下去

```
while <条件>:
    循环的代码
```

写一个死循环

```
i = 0
while True:
    i = i + 1
    print(i)
```

- 为了避免死循环，需要对while的成立条件做判断，或者在循环中加入中断代码

```
# 1+2+3+....+100=?
i = 100
sum = 0
while i:
    sum += i
    i -= 1
print(sum)
```

```
i = 0
sum = 0
while True:
    sum += i
    i += 1
    if i == 101:
        break       # 退出循环
print(sum)
```

`continue`可以跳过本次循环

```
i = 0
sum = 0
while True:
    i += 1
    if i % 2 == 0:
        continue
        # 当遇到continue的时候跳过本次循环后续代码
    if i == 101:
        break
    sum += i
print(sum)
```

问题：100以内的斐波那契数列之和

字符串的操作

- 字符串按照字符从0开始编号，这个编号作为索引，可以在读物字符串的时候进行切片

```
word = "1234567890"
print(word[0])
print(word[0:3])    # 顾头不顾尾
print(word[0:])
print(word[0::2])
print(word[-1::-1])
# 从后往前切，一定要加步长
print(word[-1:-5:-1])
```

- 字符串常用方法

```
========大小写相关操作===========
words = 'beautiful Is better Than ugly.'
print(words.capitalize())   # 首字母大写
print(words.upper())    # 全部大写
print(words.lower())    # 全部小写
print(words.swapcase())     # 大小写反转
print(words.title())        # 每个单词首字母大写

=========排版操作==============
words = "test"
ret = words.center(20,"*")  # 20格宽剧中，用*填充
print(ret)

=========判断字符串=============
```

```
words = 'beautiful Is better Than ugly.'
print(words.startswith("b"))    # 判断以开头
print(len(words))        # 查看文本的字符长度
print(words.endswith("."))      # 判断以结尾
print(words.find('e1t'))    # 判断是否存在，如果不存在返回-1
print(words.index("ett"))   # 判断是否存在，如果不存在报错
print(words.isdigit())      # 判断是否只有数字
print(words.isalpha())      # 判断是否只有字母
print(words.isalnum())      # 判断是否只有字母和数字

=========字符串的操作============
name = input("输入名字: ")
print(name.strip("* "))     # 使用strip("需要去除的")去除两头的东西
name = input("你可以输入多个用户名，用,分隔: ")
print(name.split(","))      # 将字符串用指定符号进行分隔
print(name.replace("旧字符串","新字符串"))      # 替换字符串
```

- 字符串的格式化，编辑字符串模板，允许之后替换字符串中特定的字符

```
name = input("姓名")
sex = input("性别")
hobby = input("爱好")
info = '''=====%s个人信息=====
姓名: %s
性别: %s
爱好: %s
''' % (name,name,sex,hobby)
print(info)
```

```
name = input("姓名")
sex = input("性别")
hobby = input("爱好")
info = '''====={name}个人信息=====
姓名: {name}
性别: {sex}
爱好: {hobby}
'''.format(name=name,sex=sex,hobby=hobby)
print(info)
```

列表

- 列表是用来装其他数据的容器，在列表中可以有多个元素，这些元素可以是任何数据类型

```
# 列表的定义
li = [1,'hello world',[23,123,'asd']]
print(li,type(li))
```

- 列表中的数据可以有如下方式增加

```
nameli = []
while True:
    name = input("输入用户名: ")
    if name == 'q':
        break
    nameli.append(name)     # 将数据加到列表的最后
print(nameli)

li = ['aa', 'bb', 'cc', 1, 2, 3]
li.insert(3,"dd")       # 按照索引去添加数据
print(li)
```

- 列表中元素的删除

```
li = ['aa', 'bb', 'cc', 1, 2, 3]
print(li.pop(1))       # 按照索引去删除列表
print(li.pop())         # 删除最后一个元素
print(li)
```

- 列表中元素的修改

```
li = ['aa', 'bb', 'cc', 1, 2, 3]
li[2]='bb'
print(li)
```

- 可以通过循环来查找列表中的元素

```
li = ['aa', 'bb', 'cc', 1, 2, 3]

for i in li:
    print(i)   # 循环的取值
```

元组

可以看作不可修改的列表

字典

- 是用来存储键值对，通过键查找值速度极快
- 键必须是不可修改数据类型：元组，字符串，整数，布尔值
- 值可以是任何数据类型
- 字典在python3.5以前是无序的，3.6是按照创建的顺序显示，3.7以后字典是有序的

```
dic1 = {"name":"cs", "age":18, "hobby":"男"}
print(dic1['name'])
```

- 字典使用如下方式增加元素

```
dic1 = {"name":"cs", "age":18, "hobby":"男"}
dic1["sex"] = "男"   # 增加元素
dic1.setdefault('sex','女')  # 无则添加，有则不动
print(dic1)
```

- 字典使用如下方式删除元素

```
dic1 = {"name":"cs", "age":18, "hobby":"男"}
print(dic1.pop('age'))
print(dic1)
```

- 字典可以通过如下方式查找元素

```
dic1 = {"name":"cs", "age":18, "hobby":"男"}
print(dic1.get("name1","查无此项"))
```

小练习

```
# 如果用户输入的字符串中出现了['苍老师', '小泽玛利亚', '东京热']
# 替换为等长的*,显示出来

li = ['苍老师', '小泽玛利亚', '东京热']
text = input("请输入内容: ")
for i in li:
    if i in text:
        text = text.replace(i, '*'*len(i))
print(text)
```

文件操作

- 使用python可以对文本或者二进制的文件进行读写操作
- python读取文件的流程
    - 打开文件，等到文件句柄
    - 通过句柄对文件进行操作
    - 关闭文件

```
f = open('db')      # 打开文件，得到文件句柄
data = f.read()     # 读取文件内容
print(data)         # 显示文件内容
f.close()           # 关闭文件
```

- 如果忘记关闭文件，那么f变量会一直存在于程序运行周期的内存中，这个过程中文件一直是占用状态
- 建议使用with关键字打开文件，这样就可以自动关闭了

```
with open('db') as f:
    data = f.read()
print(data)
```

文件的打开模式

r	只读打开（默认的模式）
w	只写模式，并且是覆盖
a	追加只写模式

```
with open('db', 'r', encoding="utf-8") as f:    # 使用utf-8编码只读db文件
    data = f.read()
print(data)
```

```
with open('db', 'w', encoding="utf-8") as f:    # 使用utf-8编码覆盖db文件
    data = "hello world"
    f.write(data)
```

```
with open('db', 'a', encoding="utf-8") as f:    # 使用utf-8编码追加db文件中的内容
    data = "hello world"
    f.write(data)
```

- 如果读取或者写入的文件不是文本文档，那么就只能二进制方式操作

rb	只读打开（默认的模式），读取二进制
wb	只写模式，并且是覆盖，写入二进制
ab	追加只写模式，追加二进制

‘+’模式

r+	可读可写，如果文件不存在，就会报错
w+	可写可读，如果文件不存在，就会创建
a+	可追加可读，如果文件不存在，就会创建

seek光标移动

```
with open('db', 'w+', encoding="utf-8") as f:
    data = "hello python"
    f.write(data)
    f.seek(0)       # 光标移动到文本一开始
    data2 = f.read()
print(data2)
```

今日作业

作业提示：

```python
import time
a = time.time()     # 1970年1月1日0点到现在过去了多少秒
print(a)
```

作业内容：

```python
# 1.提供登录功能，三次登录错误就退出
# 2.提供注册功能
#       a. 对用户名做限制，不允许少于4位，不允许数字开头与纯数字用户名，并且去除用户名前后空格
#       b. 对密码做限制，不允许少于6位，必须同时有数字和字母
# 3. 注册完成后，将用户名和密码存放于文件中
#       a. 注册除了第2题的功能外，还需要阻止相同的用户名存在
#       b. 三次密码错误，就锁定10秒，需要import time
```

```python
# A,B,C,D,E五个人捕鱼，不计其数，然后休息，
# 早上A第一个醒来，将鱼均分成五份，把多余的一条鱼扔掉，拿走自己的一份，
# B第二个醒来，也将鱼均分为五份，把多余的一条鱼扔掉，拿走自己的一份。
# CDE依次醒来，也按同样的方法拿鱼，问他们合伙至少捕了几条鱼。
```