笔记

> 版权：智泊AI
>
> 作者：Jeff

一、os模块

> os 用于获取系统的功能，主要用于操作文件或者文件夹
>
> 代码演示：

```
import os

# listdir  查看指定目录下面所有的文件夹和文件
os.listdir(r"c:/jeff")

# getcwd()  获取当前路径
os.getcwd()

# mkdir()   创建文件夹 (不能创建已经存在的文件夹)
os.mkdir("demo")

# makedirs()  创建多层文件夹
os.makedirs("a/b/c")

# rmdir()  删除文件夹 (只能删除空文件夹)
os.rmdir("demo")

# rename()  重命名文件夹或者重命名文件
os.rename("a", "a11")
# ./表示当前目录   ../表示上级目录
os.rename("../test.py","../demo.py")

# remove() 删除文件
os.remove("demo.py")


# os.path.join()   拼接路径
os.path.join(r"c:/jeff", "1.py")

# os.path.split()  拆分路径
path = r"c:/jeff/1.py"
os.path.split(path)

# os.path.splitext()  拆分文件和扩展名
os.path.splitext(path)

# os.path.abspath 获取绝对路径
```

```
os.path.abspath("1.py")

# os.path.getsize()   获取文件大小
os.path.getsize("1.py")

# os.path.isfile()  判断是否是文件,若是文件返回True 若不是文件 返回False
os.path.isfile("1.py")    # True

# os.path.isdir()   判断是否是文件夹, 若是文件夹 返回True 若不是文件夹 返回False
os.path.isdir("a11")  # True

# os.path.exists()  判断文件或者文件夹是否存在   若存在返回True   若不存在 返回False
os.path.exists("demo.py")   #False


# 功能总结
'''
1.os.listdir()     获取指定路径下的文件夹和文件    (是一个列表)
2.os.mkdir()       创建目录(目录存在,不能创建)
3.os.makedirs()    创建多层目录
4.os.rmdir()       删除目录
5.os.remove()      删除文件
6.os.rename()      重命名文件或者重命名文件夹


 7.os.path.join()    拼接路径
 8.os.path.split()   拆分路径
 9.os.path.splitext()   拆分文件名和扩展名
 10.os.path.isfile()   判断是否是文件
 11.os.path.isdir()    判断是否是目录
 12.os.path.exists()   判断文件或者文件夹是否存在
  13.os.path.getsize() 获取文件大小
 '''
```

二. 时间模块

1 time时间模块【掌握】

> 代码演示:

```
import time

# 获取时间戳   从1970年1月1日0时0分0秒到现在经过的秒数
time.time()

# 延迟程序多长时间执行一次
time.sleep()
```

2 datetime日期模块【掌握】

> 是对time模块的封装，比time模块更加全面

```
import datetime

# 获取当前的日期对象
date = datetime.datetime.now()
print(date)

# 设置日期对象
d = datetime.datetime(year=2026,month=8,day=16,hour=10,minute=23,second=11)
print(d)
print(type(d))
print(d.year,d.month,d.day)  # 年  月   日
print(d.hour,d.minute,d.second) # 时  分   秒
print(d.date())
print(d.time())

# 将datetime.datetime类型转换为字符串

# strftime() 将日期对象转换为字符串
print(type(d.strftime("%Y-%m-%d %H:%M:%S")))
print(d.strftime("%Y{}%m{}%d{}").format("年","月","日"))

# strptime() 将字符串转换为日期对象
str1 = "2026-08-15 10:40:21"
print(type(datetime.datetime.strptime(str1,'%Y-%m-%d %H:%M:%S')))

# timestamp()  日期对象转换为时间戳
# fromtimestamp()  时间戳转换为日期对象
print(datetime.datetime.fromtimestamp(1823046991.0))

# 时间差
d1 = datetime.datetime(2026,4,13)
d2 = datetime.datetime(2025,4,1)
print(d1 - d2)
print(d2 - d1)

# timedelta   代表两个日期之间的时间差
dt = datetime.timedelta(days=5,hours=8)
print(d1 + dt)
print(d1 - dt)

'''
# %y 两位数的年份表示（00-99）
# %Y 四位数的年份表示（0000-9999）
# %m 月份（01-12）
# %d 月内中的一天（0-31）
```

```
# %H 24小时制小时数（0-23）
# %I 12小时制小时数（01-12）
# %M 分钟数（00-59）
# %S 秒（00-59）
# %a 本地简化星期名称
# %A 本地完整星期名称
# %b 本地简化的月份名称
# %B 本地完整的月份名称
# %c 本地相应的日期表示和时间表示
# %j 年内的一天（001-366）
# %p 本地A.M.或P.M.的等价符
# %U 一年中的星期数（00-53）星期天为星期的开始
# %w 星期（0-6），星期天为星期的开始
# %W 一年中的星期数（00-53）星期一为星期的开始
# %x 本地相应的日期表示
# %X 本地相应的时间表示
# %% %号本身
'''
```

三. json模块

```
# json解析：字符串 => 字典
import json

s = '{"name": "ikun", "age": 26}'
print(type(s))

d = json.loads(s)
print(d, type(d))  # dict {'name': 'ikun', 'age': 26}
print(d['name'])

# json序列化
d = {'name': 'ikun', 'age': 26}
s = json.dumps(d)
print(s, type(s))
```

四. 文件操作

文件读写

1.概念

> 在Python中，通过打开文件生成一个文件对象【文件描述符】操作磁盘上的文件，操作主要有文件读写

2 读文件

> 操作步骤：
>
> a.打开文件：open（）
>
> b.读取文件内容：read()
>
> c.关闭文件:close()
>
> 说明：最后一定不要忘了文件关闭，避免系统资源的浪费【因为一个文件对象会占用系统资源】
>
> 代码演示：

```
# 一、打开文件
"""
open(path,flag[,encoding,errors])
path:指定文件的路径【绝对路径和相对路径】
flag:打开文件的方式
   r:只读、
   rb:read binary,以二进制的方式打开，只读【图片，视频，音频等】
   w:只能写入
   wb:以二进制的方式打开，只能写入【图片，视频，音频等】
   a:append,如果一个文件不为空，当写入的时候不会覆盖掉原来的内容

encoding：编码格式：utf-8

"""
path = r"a.txt"

#调用open函数，得到了文件对象
f = open(path, "r", encoding="utf-8")

# 二、读取文件内容
# 1.读取全部内容
s = f.read()
print(s)

# 2.读取指定的字符数
str1 = f.read(2)
print(str1)
str1 = f.read(2)
print(str1)
str1 = f.read(2)
print(str1)


# 3.读取整行，不管该行有多少个字符
```

```
str2 = f.readline()
print(str2)
str2 = f.readline()
print(str2)


# 4.读取一行中的指定的字符
str3 = f.readline(3)
print(str3)

# 5.读取全部的内容，返回的结果为一个列表，每一行数据为一个元素
str4 = f.readlines()
print(str4)


# 三、关闭文件
f.close()
```

with-as写法：

```
# 读取文件的简写形式 : with - as
#   好处：可以自动关闭文件，避免忘记关闭文件导致的资源浪费

path = "a.txt"
with open(path,"r", encoding="utf-8") as f:
  result = f.read()
  print(result)
```

3 写文件

操作步骤：

a.打开文件：open()

b.写入数据：write()

c.关闭文件：close()

代码演示：

```
path = "a.txt"

# 1.打开文件
# 注意：写入文件的时候，文件可以不存在，当open的时候会自动创建文件
f = open(path, "w", encoding="utf-8")
# f = open(path, "a", encoding="utf-8")  # 追加写

# 2.写入数据
f.write("hello")
```

```python
# 3.关闭文件
f.close()


# 简写形式
with open(path,"w", encoding="utf-8") as f:
  f.write("hello")
```