阿里云开发者社区
ALIBABA CLOUD DEVELOPER COMMUNITY

Python 脚本速查手册

详解运维工程师如何用 Python 打造工具

阿里云开发者学堂推荐配套教材

藏经阁

阿里云开发者电子书系列

阿里云开发者“藏经阁”
海量电子书免费下载

钉钉扫一扫进入官方答疑群

开发者学堂【Alibaba Cloud Linux 技术图谱】
更多好课免费学

目录

前言

Python 对于运维工程师来说，是一门必须掌握的技能。Python 是运维工程师的利器，相比于 Bash ，它拥有更加完善的生态和支持，你可以使用 Python 更加简单的完成过去在 Bash 当中复杂的功能。

同时，Python 也是众多运维工具的好伙伴，Ansible、SaltStack 等技术，都是基于 Python 写就的，掌握了 Python ，你能给节省更多的时间，去做更重要的事情。

本手册内容主要分为以下几部分内容：

1. Click 语法速查：使用 Click 编写 Cli 时会用到的核心语法。

2. Python 常用运维模块：主要介绍在编写 Python 运维脚本过程中，主要使用的一些模块和基本用法。

3. Python 运维脚本案例：一些实用 Python 编写的运维脚本的案例。

Click 语法速查

不同类型的 Param 类型

str

字符串类型，传入参数会自动转换成字符串。

int

数字类型，传入参数会自动转换成数字。

float

浮点型，传入参数会自动转换为浮点数字。

bool

布尔值，传入参数会自动转换为布尔值。

click.File(mode='r', encoding=None, errors='strict', lazy=None, atomic=False)

文件类型，传入参数会被自动以文本形式读取内容。

其中，Mode 用于设定读取的模式；

click.Path(exists=False, file_okay=True, dir_okay=True, writable=False, readable=True, resolve_path=False, allow_dash=False, path_type=None)

路径类型，传入参数会自动以路径形式读取。

其中 exists 要求文件是否必须存在；file_okay 判断是否是文件；dir_okay 判断是否是目

录；writable 判断文件是否可写；readable 判断文件是否可读；

click.Choice(choices, case_sensitive=True)

选择类型，可以用于设置选择。输入内容必须是设定的选择。

```
@click.command()
@click.option('--hash-type',
              type=click.Choice(['MD5', 'SHA1'], case_sensitive=False))
def digest(hash_type):
    click.echo(hash_type)
```

click.IntRange(min=None, max=None, min_open=False, max_open=False, clamp=False)

数字范围类型，可以设定允许的数字范围。

```
@click.command()
@click.option("--count", type=click.IntRange(0, 20, clamp=True))
@click.option("--digit", type=click.IntRange(0, 9))
def repeat(count, digit):
    click.echo(str(digit) * count)
```

click.FloatRange(min=None, max=None, min_open=False, max_open=False, clamp=False)

浮点数字类型，用法类似 intRange。

click.DateTime(formats=None)

时间类型，传入数据需可被 Format，format 结果由参数定义。

不同类型的 Option

必选 option

```
@click.command()
@click.option('--n', required=True, type=int)
def dots(n):
    click.echo('.' * n)
```

多参数 option

```
@click.command()
@click.option('--pos', nargs=2, type=float)
def findme(pos):
    a, b = pos
    click.echo(f"{a} / {b}")
```

Flag 型 option

```
@click.command()
@click.option('--pos', nargs=2, type=float)
def findme(pos):
    a, b = pos
    click.echo(f"{a} / {b}")
```

包含提问的 option

```
@click.command()
@click.option('--name', prompt=True)
def hello(name):
    click.echo(f"Hello {name}!")
```

密码型 option

```
@click.command()
```

```
@click.password_option()
def encrypt(password):
    click.echo(f"encoded: to {codecs.encode(password, 'rot13')}")
```

不同类型的 Argument

普通 Argument

```
@click.command()
@click.argument('filename')
def touch(filename):
    """Print FILENAME."""
    click.echo(filename)
```

文件 Argument

```
@click.command()
@click.argument('input', type=click.File('rb'))
@click.argument('output', type=click.File('wb'))
def inout(input, output):
    """Copy contents of INPUT to OUTPUT."""
    while True:
        chunk = input.read(1024)
        if not chunk:
            break
        output.write(chunk)
```

文件路径 Argument

```
@click.command()
@click.argument('filename', type=click.Path(exists=True))
def touch(filename):
```

```
    """Print FILENAME if the file exists."""
    click.echo(click.format_filename(filename))
```

特殊用法

命令行输出到标准输出

```
import click


click.echo('Hello World!')
```

命令行输出彩色

以下两种方法均可：

```
import click


click.echo(click.style('Hello World!', fg='green'))
click.echo(click.style('Some more text', bg='blue', fg='white'))
click.echo(click.style('ATTENTION', blink=True, bold=True))
```

或

```
import click


click.secho('Hello World!', fg='green')
click.secho('Some more text', bg='blue', fg='white')
click.secho('ATTENTION', blink=True, bold=True)
```

命令行输出分页

```
@click.command()
def less():
```

```
    click.echo_via_pager("\n".join(f"Line {idx}" for idx in range(200)))
```

清空屏幕

```
import click
click.clear()
```

命令行进度条

```
import click


with click.progressbar(all_the_users_to_process) as bar:
    for user in bar:
        modify_the_user(user)
```

Python 常用运维模块

OS 模块

OS 模块是运维工程师在运维工作过程中使用最为频繁的模块，其中提供了大量操作系统底层的借口，帮助运维工程师完成自己的运维工作。

os.getcwd()

获取当前工作目录，即当前 Python 脚本的目录路径。

os.chdir("dirpath")

更改当前脚本工作目录，相当于在 Shell 下执行 cd 命令。

os.curdir

返回当前目录路径。

os.pardir

返回上一级目录路径。

os.makedirs("dir1/dir2")

递归生成目录路径。

os.removedirs('dirname1')

删除目录，如果其中有文件，则递归删除。

os.mkdir("dir")

创建目录，不支持递归生成目录。

os.rmdir("dir")

删除目录，若目录不为空则无法删除，报错；相当于 shell 中 rmdir。

os.listdir("dir")

列出特定目录下文件和子目录。

os.remove()

删除某个特定文件。

os.rename("old name","newname")

重命名某个文件。

os.stat("filename")

获取某个文件/目录的信息。

os.sep

输出目录路径分隔符，可以用于处理不同平台之间的差异。

os.linesep

输出文件行终止符，可以用于处理不同平台之间的差异。

os.pathsep

输出用于分隔文件路径的字符串，可以用于处理不同平台之间的差异。

os.name

输出当前操作系统的名称，可以用于处理不同平台之间的差异。

os.environ

获取当前系统的环境变量。

os.access(path,mode)

判断是否有权限对文件进行读、写、执行等操作。

os.chmod(path,mode)

修改文件的权限。

os.chown(path,uid,gid)

修改文件的属主和属组。

os.symlink(src,dst)

创建软连接。

os.times()

获取系统进程运行时间。

Shutil 模块

Shutil 模块提供了一系列的 Shell 功能，让运维工程师可以以更简单的方式完成 OS 模块无法完成的功能。

shutil.copyfile(src,dst)

复制源文件到目标路径。

shutil.copymode(src,dst)

复制源文件权限到目标文件。

shutil.copystat(src,dst)

复制源文件文件、最近修改时间、Flag 等到目标文件。

shutil.copy(src,dst)

复制源文件到目标路径。

shutil.copy2(src,dst)

复制源文件及相关元信息到目标路径。

shutil.move(src,dst)

移动文件/目录。

shutil.disk_usage(path)

返回对应路径的磁盘的使用统计情况。

shutil.copytree(src,dst)

将 src 的整个目录树复制到 dst 目录。

shutil.which

获取要执行的命令的路径。

Sys 模块

sys.exit(0)

退出当前进程。参数为 0 时表示正常退出；参数为 1 时表示异常退出。

sys.version

获取 Python 版本。

sys.path

获取模块的搜索路径。

sys.platform

获取操作系统名称。

sys.stdin

标准输入。

sys.stdout

标准输出。

sys.stderr

错误输出。

Datetime 模块

datetime.date(year,month,day)

构建一个 date 对象。

datetime.date(year,month,day).today()

返回当前的本地日期。

datetime.date(year,month,day).strftime(format)

对时间进行格式化后输出。

datetime.now()

返回当前时间。

datetime.timestamp()

返回当前时间戳。

psutil 模块

psutil 模块非 Python 内置模块，在使用前需要执行 pip install psutil 进行安装。

psutil.cpu_count()

获取 CPU 逻辑核心数。

psutil.cpu_count(logical=False)

获取 CPU 物理核心数。

psutil.cpu_times()

获取 CPU 的用户/系统/空闲时间。

psutil.virtual_memory()

获取系统的内存信息。

psutil.swap_memory()

获取系统交换内存的信息。

psutil.disk_partitions()

获取磁盘分区信息。

psutil.disk_usage('/')

获取磁盘使用情况。

psutil.disk_io_counters()

获取磁盘 IO。

psutil.net_io_counters()

获取网络读写包的情况。

psutil.net_if_addrs()

获取网络接口信息。

psutil.net_if_stats()

获取网络接口状态。

psutil.pids()

获取所有进程信息。

实用 Python 脚本案例

查看 CPU、内存信息

```
#!/usr/bin/env python3
# author: http://github.com/opsonly
import psutil
def memissue():
    print('内存信息：')
    mem = psutil.virtual_memory()
    # 单位换算为 MB
    memtotal = mem.total/1024/1024
    memused = mem.used/1024/1024
    membaifen = str(mem.used/mem.total*100) + '%'

    print('%.2fMB' % memused)
    print('%.2fMB' % memtotal)
    print(membaifen)

def cuplist():
    print('磁盘信息：')
    disk = psutil.disk_partitions()
    diskuse = psutil.disk_usage('/')
    #单位换算为 GB
    diskused = diskuse.used / 1024 / 1024 / 1024
    disktotal = diskuse.total / 1024 / 1024 / 1024
    diskbaifen = diskused / disktotal * 100
    print('%.2fGB' % diskused)
    print('%.2fGB' % disktotal)
    print('%.2f' % diskbaifen)
```

```
memissue()
print('********************')
cuplist()
```

统计 Nginx 日志中的 IP 并绘制成图形

```
#!/usr/bin/env python3
# author: http://github.com/opsonly
import matplotlib.pyplot as plt
#
nginx_file = 'nginx2018-12-18_07:45:26'

ip = {}
# 筛选 nginx 日志文件中的 ip
with open(nginx_file) as f:
    for i in f.readlines():
        s = i.strip().split()[0]
        lengh = len(ip.keys())

        # 统计每个 ip 的访问量以字典存储
        if s in ip.keys():
            ip[s] = ip[s] + 1
        else:
            ip[s] = 1


#以 ip 出现的次数排序返回对象为 list
ip = sorted(ip.items(), key=lambda e:e[1], reverse=True)
```

```
#取列表前十
newip = ip[0:10:1]
tu = dict(newip)

x = []
y = []
for k in tu:
    x.append(k)
    y.append(tu[k])
plt.title('ip access')
plt.xlabel('ip address')
plt.ylabel('PV')

#x 轴项的翻转角度
plt.xticks(rotation=70)

#显示每个柱状图的值
for a,b in zip(x,y):
    plt.text(a, b, '%.0f' % b, ha='center', va= 'bottom',fontsize=7)



plt.bar(x,y)
plt.legend()
plt.show()
```