Python

Snippet list

Edit file

The following functions can be used to edit any provided file. In the selected file a string can be searched that can then be replaced with a given string.

Edit a file with python

Show/Hide

```
# search file for string
def searchLineWithString(self, string, file):
    for num, line in enumerate(file, 1):
        if string in line:
            return num - 1

# edit line dependend on provided parameter
def editLine(self, file, searchString, newLine):
    selectedFile = open(file, "r")
    readSelectedFile = selectedFile.readlines()
    selectedFile.close()
    # find specific line
    lineToEdit = searchLineWithString(self, searchString, readSelectedFile)
    # edit required line
    readSelectedFile[lineToEdit] = newLine

    # write lines to files
    selectedFile = open(file, "w")
    selectedFile.writelines(readSelectedFile)
    selectedFile.close()
```

Send mail

The following functions can be used to emails from python.

Send mails with python

Show/Hide

```
    EMAILRECEIVER = ""
    EMAILSENDER = ""
    MAILSERVER = ""
    MAILSERVERPORT = ""
    EMAILSENDER = ""

    def buildMail(self):
        mailText = ""
        # build your mail here string by string (mailText += "")
        return mailText

    def sendMail(self):
        if EMAILRECEIVER and EMAILSENDER and MAILSERVER and MAILSERVERPORT:
            import smtplib
            from email.mime.multipart import MIMEMultipart
            from email.mime.text import MIMEText
            from email.header import Header

            smtp = smtplib.SMTP()
            smtp.connect(MAILSERVER, MAILSERVERPORT)
            # define subject here
        subject = ""
            msgRoot = MIMEMultipart("alternative")
            msgRoot['Subject'] = Header(subject, "utf-8")
            msgRoot['From'] = "Python-Script <" + EMAILSENDER + ">"
            msgRoot['To'] = EMAILRECEIVER
            mailContent = MIMEText(buildMail(self))
            mailText = MIMEText(mailContent, "plain", "utf-8")
            msgRoot.attach(mailText)
            try:
                smtp.sendmail(EMAILSENDER, EMAILRECEIVER,
msgRoot.as_string())
            except:
                print(self, "Failed to send mail to " + EMAILRECEIVER + " |
Check your settings!")
                print(traceback.format_exc())

        else:
            print(self, "Mailing disabled or not configured properly.")
```

Send mails with python using AUTH

Show/Hide

```
    # MAILCONFIG
    MAILSERVER = ""
    MAILSERVERPORT = 587
    MAILUSER = ""
    MAILPASSWORD = ""
    EMAILRECEIVER = ""
    EMAILSENDER = ""

    def buildMail(self):
        mailText = ""
        # build your mail here string by string (mailText += "")
        return mailText

    def sendMail(self):
        if EMAILRECEIVER and MAILUSER and MAILSERVER and MAILSERVERPORT:
            import smtplib
            from email.mime.multipart import MIMEMultipart
            from email.mime.text import MIMEText
            from email.header import Header

            smtp = smtplib.SMTP(MAILSERVER)
            smtp.connect(MAILSERVER, MAILSERVERPORT)
            smtp.ehlo()
            smtp.starttls()
            smtp.ehlo()
            smtp.login(MAILUSER, MAILPASSWORD)
            # define subject here
        subject = ""
            msgRoot = MIMEMultipart("alternative")
            msgRoot['Subject'] = Header(subject, "utf-8")
            msgRoot['From'] = EMAILSENDER
            msgRoot['To'] = EMAILRECEIVER


            mailContent = buildMail(self)
            mailText = MIMEText(mailContent, "plain", "utf-8")

            msgRoot.attach(mailText)
            try:
                smtp.sendmail(MAILUSER, EMAILRECEIVER, msgRoot.as_string())

            except:
                print(self, "Failed to send mail to " + EMAILRECEIVER + " |
Check your settings!")
                print(self, traceback.format_exc())

        else:
            print(self, "Mailing disabled or not configured properly.")
```

Execute Bash command and capture output

The output will be captured when the command exited.

Run subprocess

Show/Hide

```
resultEncoded = subprocess.run("/command/to/execute", capture_output=True,
shell=True)
result = resultEncoded.stdout.decode()[:-1]
resultErr = resultEncoded.stderr.decode()[:-1]
```

Sourced from stackoverflow.com - Python Email

Get current Memory (RAM) information/statistics (Linux)

Using `op.popen` and Linux build-in binary `free`.

Get memory information on Linux

Show/Hide

```
total_memory, used_memory, free_memory, shared_memory, cached_memory,
available_memory = map(int, os.popen('free -t -
m').readlines()[1].split()[1:])
```