Tips

?? UTF-8 ??

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

Find all installed modules

```
help("modules");
```

Virtual Environment

- How to Install and Manage Python Versions in Linux - Make Tech Easier

Conda

```
# Create a virtual env
conda create -n myproj python=3.11

# Activate the virtual env
conda activate myproj

# Deactivate the virtual env
conda deactivate
```

Python 3.4+ built-in venv

```
# Install venv
sudo apt install python3-venv

# Enable venv
mkdir myproject
cd myproject
python -m venv .venv
```

```
# Activate the venv
source .venv/bin/activate

# Delete the venv
deactivate
rm -rf .venv

# Change the App directory after activating venv
cd /path/to
mv old new
cd new/.venv/bin
old_path="/path/to/old/.venv"
new_path="/path/to/new/.venv"
find ./ -type f -exec sed -i "s|$old_path|$new_path|g" {} \;
cd /path/to/new
source .venv/bin/activate
```

With virtualenv and virtualenvwrapper

```
# Installing virtualenv and virtualenvwrapper
sudo pip install virtualenv virtualenvwrapper

# Update the profile ~/.bashrc
# Add the  following lines

# Python virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh

# Reload the profile
source ~/.bashrc

# Creating python virtual environment
# The py3cv3 is a self-defined name
mkvirtualenv py3cv3 -p python3

# Enter the specified virtual environment
workon py3cv3
```

```
# Exit the the specified virtual environment
deactivate

# List all of the environments.
lsvirtualenv

# Remove an environment
rmvirtualenv py3cv3
```

Timestamp

```
timestamp = datetime.datetime.now()
print("It is {}".format(timestamp.strftime("%A %d %B %Y %I:%M:%S%p")))
```

Math

```
total += 1
```

If-else

```
# Boolean, none
if motion is not None:
if not flag:

# Number
if delay > 0:
if delay == 0:
if total > frameCount:

# String
if "blue" in style:
if authors.startswith('['):
    authors = authors.lstrip('[').rstrip(']')

# One-liner
def doi_url(d): return f'http://{d}' if d.startswith('doi.org') else f'http://doi.org/{d}'

# Multiple conditions
temperature = 25
```

```
if temperature > 30:
    print('Hot')
elif temperature > 20 and temperature <= 30:
    print('Warm')
else:
    print('Cool')

# Reverse the True
temperature = 15
if not temperature > 20:
    print('Cool')
#
temperature = 25
humidity = 55
rain = 0
if temperature > 30 or humidity < 70 and not rain > 0:
    print('Dry conditions')

# Logical operators, AND, OR, NOT
if status >= 200 and status <= 226:
if status == 100 or status == 102:
if not(status >= 200 and status <= 226):
```

operator

operator	use
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
==	equal to
!=	not equal to

Command Arguments

- Manual: https://docs.python.org/3/library/argparse.html

```
import argparse
# construct the argument parser and parse the arguments
```

```
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--interval", required=False,
	help="Seconds to Interval (Default:30)", default="30", type=int)
ap.add_argument("-o", "--output", required=False,
	help="Path to Output Logs (Default:std-out)")
ap.add_argument("mac",
	help="MAC address of LYWSD02 device", nargs="+")
args = vars(ap.parse_args())

# Usage
intv = args["interval"]
logfile = args["output"]
```

```
from argparse import ArgumentParser

def _get_args():
    parser = ArgumentParser()
    parser.add_argument("-c", "--checkpoint-path", type=str, default=DEFAULT_CKPT_PATH,
                        help="Checkpoint name or path, default to %(default)r")
    parser.add_argument("--cpu-only", action="store_true", help="Run demo with CPU only")

    parser.add_argument("--share", action="store_true", default=False,
                        help="Create a publicly shareable link for the interface.")
    parser.add_argument("--inbrowser", action="store_true", default=False,
                        help="Automatically launch the interface in a new tab on the default browser.")
    parser.add_argument("--server-port", type=int, default=8000,
                        help="Demo server port.")
    parser.add_argument("--server-name", type=str, default="127.0.0.1",
                        help="Demo server name.")

    args = parser.parse_args()
    return args

def _test_args(args);
    if args.cpu_only:
        device_map = "cpu"
    else:
        device_map = "auto"

    ckp_path = args.checkpoint_path
```

```
    return device_map, ckp_path

def main():
    args = _get_args()
    device_map, ckp_path = _test_args(args)

if __name__ == '__main__':
    main()
```

```
#
# Nagios2 HTTP proxy test
#
# usage: check_http_proxy --proxy=proxy:port --auth=user:pass --url=url --timeout=10 --warntime=5 --
expect=content

import sys
import getopt

def get_cmdline_cfg():
    try:
        opts, args = getopt.getopt(
            sys.argv[1:],
            "p:a:t:w:e:u:",
            ["proxy=", "auth=", "timeout=", "warntime=", "expect=", "url="]
        )
    except getopt.GetoptError, err:
        print("SCRIPT CALLING ERROR: {0}".format(str(err)))

    ### Build cfg dictionary
    cfg = {}
    for o, a in opts:
        if o in ("-p", "--proxy"):
            cfg["proxy"] = a
        elif o in ("-a","--auth"):
            cfg["auth"] = a
        elif o in ("-t","--timeout"):
            cfg["timeout"] = float(a)
        elif o in ("-w","--warntime"):
```

```
        cfg["warntime"] = float(a)
    elif o in ("-e","--expect"):
        cfg["expect"] = a
    elif o in ("-u","--url"):
        cfg["url"] = a

    # These are required
    for req_param in ("url", "proxy"):
        if req_param not in cfg:
            print("Missing parameter: {0}".format(req_param))

    return cfg

# Usage
if __name__ == '__main__':
    cfg = get_cmdline_cfg()

    if "auth" in cfg:
        proxy_url = "http://{auth}@{proxy}/".format(**cfg)
    else:
        proxy_url = "http://{proxy}/".format(**cfg)
```

Import/Write files

```
# Read a txt file
with open("update_log.txt", "r") as file:
    updates = file.read()

print(updates)

# Write a txt file
# With both "w" and "a", you can use the .write() method
# "a" if you want to append to a file
line = "jrafael,192.168.243.140,4:56:27,True"
with open("access_log.txt", "w") as file:
    file.write(line)

# Write a CSV or multi-lines file
login_file = """username,ip_address,time,date
```

```
tshah,192.168.92.147,15:26:08,2022-05-10
dtanaka,192.168.98.221,9:45:18,2022-05-09
tmitchel,192.168.110.131,14:13:41,2022-05-11
daquino,192.168.168.144,7:02:35,2022-05-08
eraab,192.168.170.243,1:45:14,2022-05-11
jlansky,192.168.238.42,1:07:11,2022-05-11
acook,192.168.52.90,9:56:48,2022-05-10
"""

with open("login.txt", "w") as file:
    file.write(login_file)
```

Revision #54
Created 9 November 2022 14:33:15 by Admin
Updated 23 September 2024 13:57:27 by Admin