Tips

?? UTF-8 ??

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

Find all installed modules

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

???????????

```
import powerline
powerline.__path__


# Return ['/home/alang/.local/lib/python3.10/site-packages/powerline']
```

Print

- `print( ... , end=" ")` ???????????
- `print("[" + str(left) + "|")` ????????????? `str()` ??
- `print()` ??????????
- `print(, file=sys.stderr)` : ????????? `sys.stdout` (????)

```
for left in range(7):
  for right in range(left, 7):
    print("[" + str(left) + "|" + str(right) + "]", end=" ")
  print()
```

Print the List with `join()`

```
greetings = ["Hello", "world"]
print(" ".join(greetings))  # Prints "Hello world"
```

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

sys.argv

- ??? Script ????

```
import sys

logfile = sys.argv[1]
with open(logfile) as f:
  for line in f:
    if "CRON" not in line:
      continue
    print(line.strip())
```

argparse

- ??? Script ????
- 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)
```

Reading and Writing files

Open mode

- r : Read only (default)
- w : Write only
- a : Append
- r+ : Read-Write
- t : Text mode (default)
- b : Binary mode
- x : open for exclusive creation, failing if the file already exists

Read file: ???????????? String ??

>

> Tip:? with ??????????????

```
with open("spider.txt") as file:
    for line in file:
        print(line.strip().upper())
```

Read file: ?????????????? List ??

```
file = open("spider.txt")
lines = file.readlines()
file.close()
lines.sort()
print(lines)
```

Write a file: ????? String ???????????? string ?????

```
with open("novel.txt", "w") as file:
    file.write("It was a dark and stormy night")

# Return 30
# when successful, return the length of the string
```

```
guests = open("guests.txt", "w")
initial_guests = ["Bob", "Andrea", "Manuel", "Polly", "Khalid"]

for i in initial_guests:
    guests.write(i + "\n")

guests.close()
```

Read and Write file

```
# 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)
```

Encoding: ??????????????????

```
f = open('workfile', 'w', encoding="utf-8")

with open('log_file', mode='r',encoding='UTF-8') as file:
    for log in file.readlines():
```

File and Directory

Managing files

```
import os
os.remove("novel.txt")

os.rename("first_draft.txt", "finished_masterpiece.txt")

os.path.exists("finished_masterpiece.txt")
# Return True or False

os.path.getsize("spider.txt")
#This code will provide the file size
```

```
import datetime
timestamp = os.path.getmtime("spider.txt")
datetime.datetime.fromtimestamp(timestamp)
#This code will provide the date and time for the file in an
#easy-to-understand format

os.path.abspath("spider.txt")
#This code takes the file name and turns it into an absolute path
```

Managing directories

```
os.mkdir("new_dir")
#The os.mkdir("new_dir") function creates a new directory called new_dir

os.chdir("new_dir")
os.getcwd()
#This code snippet changes the current working directory to new_dir.
#The second line prints the current working directory.

os.mkdir("newer_dir")
os.rmdir("newer_dir")
#This code snippet creates a new directory called newer_dir.
#The second line deletes the newer_dir directory.

import os
os.listdir("website")
#This code snippet returns a list of all the files and
#sub-directories in the website directory.

dir = "website"
for name in os.listdir(dir):
    fullname = os.path.join(dir, name)
    if os.path.isdir(fullname):
        print("{} is a directory".format(fullname))
    else:
        print("{} is a file".format(fullname))
```

Using os module

```
# Create a directory and move a file from one directory to another
# using low-level OS functions.

import os

# Check to see if a directory named "test1" exists under the current
# directory. If not, create it:
dest_dir = os.path.join(os.getcwd(), "test1")
if not os.path.exists(dest_dir):
 os.mkdir(dest_dir)



# Construct source and destination paths:
src_file = os.path.join(os.getcwd(), "sample_data", "README.md")
dest_file = os.path.join(os.getcwd(), "test1", "README.md")



# Move the file from its original location to the destination:
os.rename(src_file, dest_file)
```

Using pathlib module

```
# Create a directory and move a file from one directory to another
# using Pathlib.

from pathlib import Path

# Check to see if the "test1" subdirectory exists. If not, create it:
dest_dir = Path("./test1/")
if not dest_dir.exists():
  dest_dir.mkdir()

# Construct source and destination paths:
src_file = Path("./sample_data/README.md")
dest_file = dest_dir / "README.md"

# Move the file from its original location to the destination:
src_file.rename(dest_file)
```

os.environ

- `.copy()` : ??????????? dictiorary
- .get(NAME, "") : ?? NAME ????
- `my_env["PATH"]` : ?? PATH ???

```
import os
import subprocess

my_env = os.environ.copy()
my_env["PATH"] = os.pathsep.join(["/opt/myapp/", my_env["PATH"]])

result = subprocess.run(["myapp"], env=my_env)
```

```
import os
print("HOME: " + os.environ.get("HOME", ""))
print("SHELL: " + os.environ.get("SHELL", ""))
print("FRUIT: " + os.environ.get("FRUIT", ""))
```

input

- input() : ?? string ????

```
def to_seconds(hours, minutes, seconds):
    return hours*3600+minutes*60+seconds

print("Welcome to this time converter")

cont = "y"
while(cont.lower() == "y"):
    hours = int(input("Enter the number of hours: "))
    minutes = int(input("Enter the number of minutes: "))
    seconds = int(input("Enter the number of seconds: "))

    print("That's {} seconds".format(to_seconds(hours, minutes, seconds)))
    print()
    cont = input("Do you want to do another conversion? [y to continue] ")

print("Goodbye!")
```

subprocess

Run system commands in Python

- subprocess ???????? script (???) ?????????????
- ??????????????????
- ????????? List ??????? ["command", "opt1", "opt2"]
- `.returncode` : ??????0 ????1 ???
- `.stderr` : ??????????????(An array of bytes)??? `decode()` ??? string ??

```
import subprocess
subprocess.run(["date"])
subprocess.run(["sleep", "2"])
result = subprocess.run(["ls", "this_file_does_not_exist"])
print(result.returncode)
print(result.stderr)
```

- `run( , capture_output=True)` : ??????????? *(python 3.7+ ??)*
- `.stdout` : ??????????????(An array of bytes)??? `decode()` ??? string ??

```
result = subprocess.run(["host", "8.8.8.8"], capture_output=True)
print(result.stdout)

# Output: b'8.8.8.8.in-addr.arpa domain name pointer dns.google.\n'

result = subprocess.run(["host", "8.8.8.8"], capture_output=True)
print(result.stdout.decode().split())
```

- `run(, env=my_env)` : ??????

```
import os
import subprocess

my_env = os.environ.copy()
my_env["PATH"] = os.pathsep.join(["/opt/myapp/", my_env["PATH"]])

result = subprocess.run(["myapp"], env=my_env)
```

- `run(, capture_output=True, text=True)` : ???????? decode

```
result_run = subprocess.run(['echo', 'Hello, World!'], capture_output=True, text=True)
result_run.stdout.strip()  # Extracting the stdout and stripping any extra whitespace

# Output: 'Hello, World!'
```

- `check_call()` : ??????????

```
return_code_check_call = subprocess.check_call(['echo', 'Hello from check_call!'])
print(return_code_check_call)

# Output 0
```

- `check_output()` : ???????????

```
output_check_output = subprocess.check_output(['echo', 'Hello from check_output!'], text=True)
output_check_output.strip()  # Extracting the stdout and stripping any extra whitespace

# Output 'Hello from check_output!'
```

- `Popen()` : ???????????? input/output/error ???????
- `.poll()` : ??? NONE?????????

```
process_popen = subprocess.Popen(['echo', 'Hello from popen!'], stdout=subprocess.PIPE,
text=True)
output_popen, _ = process_popen.communicate()
output_popen.strip()  # Extracting the stdout and stripping any extra whitespace

# Output: 'Hello from popen!'
```

```
process = subprocess.Popen(['sleep', '5'])
message_1 = "The process is running in the background..."

# Give it a couple of seconds to demonstrate the asynchronous behavior
import time
time.sleep(2)

# Check if the process has finished
if process.poll() is None:
message_2 = "The process is still running."
else:
```

```
message_2 = "The process has finished."

print(message_1, message_2)
```

- ?? os , Pathlib ??????????????????

```
# subprocess
subprocess.run(['mkdir', 'test_dir_subprocess2'])

# OS
os.mkdir('test_dir_os2')

# Pathlib
test_dir_pathlib2 = Path('test_dir_pathlib2')
test_dir_pathlib2.mkdir(exist_ok=True) #Ensures the directory is created only if it doesn't
already exist
```

logging

Level: DEBUG, INFO, WARNING, ERROR, CRITICAL

```
import logging

logging.warning('This is a warning message')
logging.error('This is an error message')

logging.basicConfig(level=logging.DEBUG)
logging.debug('This is a debug message')

logging.basicConfig(filename='app.log', level=logging.DEBUG)
logging.info('This message will be written to app.log')

logging.basicConfig(format='%(asctime)s - %(levelname)s - %(message)s', level=logging.DEBUG)
logging.error('This is an error with a custom format')
```

Revision #105
Created 2022-11-09 14:33:15 CST by A-Lang (Admin)
Updated 2025-12-10 09:53:52 CST by A-Lang (Admin)