Cmd Cmd Name of project

Shell in python.

Sub process library

```
Subprocess.run (["ls", "-l"])
# does not catch output.
```

Common commands

→ import Subprocess

→ Subprocess.run ("ls") # Simple the Shell

command execution

→ Subprocess.run ("dir", Shell = True) # running the
Shell built in commands.

→ Subprocess.run ('ls . -la', Shell = True)
# used to perform complex commands also

→ Subprocess.run (['ls', '-la']) # this
# provides more secqurity that -la is an argument.
# no need of spa because the space is default given.

→ p1 = Subprocess.run (['ls', '-la'])

→ print (p1)
This prints the last command status like finished or not.

→ print (P1.args)
# prints the argument passed.

→ print (P1. return code)
This prints 0 or 1 0 = This means the program cmd
# run successfully 1 = failed

→ print (p1. stdout)

this prints none because the output has been already
printed

capturing output.

→ P1 = subprocess. run (['ls', '-la'],
capture_output := True)

this captures the output and stores ~~at~~ as a binary

→ Print (P1. Stdout)

this prints the binary command output

→ Print (p1. stdout . decode ())

this decodes and prints output of the file.

→ p1 = subprocess . run (['ls', 'la'], capture_output
= True, text = True)

this returns the text form of output not Binary

→ print (p1. stdout)

now it prints the ~~org~~ output of the command.

```
DATE

→ P1 = Subprocess . run ([ 'ls', '-la' ],
  Stdout = Subprocess. ~~Pipe~~ PIPE , text= true )
→ print (p1. Stdout)
# This prints the process. Same as last command.

→ with open . ("output.txt", 'w') as f :
    → P1 = Subprocess . run ([ 'ls', '-la' ], Stdout=f,
  text = True)
  # This creates a file and store the output.
  # executing wrong command. ↓
→ p1 = Subprocess . run ([ 'ls', '-la', 'dne' ],
  capture _output = True, text = True)
  print (p1. Stderr)
  # This prints error.

  [illegible]
  print (p1. return code)
  # This returns 1 means the last command is fild

PAGE
classmate
```

DATE

→ P1 = Subprocess . run (['ls', '-la'],
Stdout = Subprocess . ~~[illegible]~~ PIPE , text= true)
→ print (p1). Stdout)
This prints the process. Same as last command.

→ with open . ("output.txt", 'w') as f :
  → P1 = Subprocess . run (['ls', '-la'], Stdout=f,
text =True)
This creates a file and store the output.
# exicuting wrong command. ↓
→ p1 = Subprocess . run (['ls', '-la', 'dne'],
capture _output =True, text = True)
print (p1. Stderr)
This prints error.

print (p1. return code)
This returns 1 means the last command is fild

PAGE

DATE

→ P1 = ~~Sub~~ Subprocess . run (['ls', '-la', 'dne'] ,
capture_ output = True , text = true, check=True)
print (p1 . Stderr)

→ p1 = Subprocess . run . (['ls', '-la', 'dne'],
Stderrs = Subprocess . DEVNULL)

→ print (p1 . Stderr)
# this line ignore the error.

using cat command to display multiple line output
→ P1 = Subprocess . run (['cat', 'test.txt'],
capture_ output = true , text = true)
→ print (P1 . Stdout)

piping 2 commands.
→ P1 = Subprocess . run (['cat', 'text.txt'],
→ capture_ ~~oup~~ output = True , text = true)
→ P2 = Subprocess . run (['grep', '-n', 'test'],
→ capture_ output = true, text = True , input =
→ P1 . Stdout)
→ Print (p2 . Stdout)

PAGE

classmate

#that was equal to cat txt 2.txt | grep -n'test'

#this can be also done as

→ P1 = Subprocess . run ('cat test.txt. | grep -n test',
capture_output = True , text = True, shell = True)

→ Print (P1. stdout)

OS process

→ import OS

#printing current working directory

→ print (OS . getcwd ())

→ OS. chdir ('location')

#this is used to change the directory

~~[illegible]~~

→ print (os. list dir ())

#used to list all directory

OS. mkdir ('os-Demo-2')

create

DATE

→ Os. mkdir ('os-demo-2')

used to create single directory

→ Os.mkdirs ('OS-Demo/2OS Demo')

used to create multiple director like dir inside

#Dir

→ Os. rmdir ('OS-Demo-2')

used to remove a single dir.

~~Os. rem~~

~~Os. removedir~~

→ Os. removedirs ('os-demo-2/Sub-Dir-1')

used to remove full directory path.

~~Print~~

→ Os. rename ('test.txt', 'demo.txt')

this is used to rename the folder,

~~Print (Os. Stat ('demo.txt', St_atime)~~

→ print (os. Stat ('demo.txt'))

used to print statues of the file or folder.

PAGE

classmate

Reading the dat modified of a file

```
- import os
- from date time import datetime
- Os. chdir ('/users/ core / Despatt')
- mod time = Os. Stat ('demo.txt'). st_mtime
- print (date time. fromtimestamp (~~modtime~~ mod_time)
```

```
~~import os~~
-> Os. chdir ('/Users/coreyschafer/Desktop')
-> for dirpath, dirnames, filename in os.walk ('/users
/coreyschafer/Destop/'):
        -> print ('current path :', dirpath)
        -> print ('Directories :', dirnames)
        -> print ('Files:', filenames)
        -> print ()
```

file curating ,

```
OS. Chdir ('/users/coreyschafer/Desktop/')
#print (os. environ. get ('home'))
```

classmate

DATE

displays the path of the file

-> file path = Os.path.join(os.environ.get('Home'),
'text.txt')

-> print (file_path)

~~with open (file path)~~

it display joing two paths.

-> print .(os.path.basename('/tmp/test.txt'))

#outputs the base object the last word here

test.txt.

-> print (Os.path.dirname ('tmp/test.txt'))

this prints the base directory here ~~for~~ tmp is base dir

-> print (os.path.split ('/tmp/test.txt'))

splits and print

-> print (os.path.exists ('/tmp/test.txt'))

#prints whether the path or file or folder exist or not.

-> print (Os.path. ~~[illegible]~~ isfile ("test.txt"))

prints whether its a file or not.

PAGE

DATE

```
- print (os.path.isdir ("[illegible]/[illegible]"))
  # Checks whether it's a directory
- print (os.path.splitext ("temp/test.txt"))
  # It gives the extension by splitting
```

DATE

→ print (os.path.is dir ('~~tmp~~ /tmp/jqhg')

# checks whether its a directory

→ print (os.path.splitext ('/tmp/text.txt'))

\#it prints the extension ~~by~~ by spliting