Python Files

---------- **FileName:**
10WindowsC.py

Content:
```
import os


var_name = os.getcwd() # the environment variable to check
if var_name in os.environ:
print(f"{var_name} exists")
else:
print(f"{var_name} does not exist")
```

---------- **FileName:**
11WindowsC.py

Content:
```
import psutil

for proc in psutil.process_iter(['pid', 'name']):
print(f"PID: {proc.info['pid']} | Name: {proc.info['name']}")
```

---------- **FileName:**
12WindowsC.py

Content:
```
import psutil
import os


for proc in psutil.process_iter(['pid', 'name']):
print(f"PID: {proc.info['pid']} | Name: {proc.info['name']}")

print("this working in windows")
Input = input("what the proccess name you want kill it: ")

process_name = Input # change this to the name you want to kill

for proc in psutil.process_iter(['pid', 'name']):
if proc.info['name'] == process_name:
print(f"Killing {process_name} with PID {proc.info['pid']}")
proc.terminate() # or proc.kill() for force kill
```

---------- **FileName:**
13WindowsC.py

Content:

```
import os

print("the code programming for Windows!!!")
print("should the app in your computer")
print("like notepad.exe or cmd or notepad")
Input = input("enter the app should the app right for the cmd: ")

os.system(Input)
```

---------- **FileName:** 14WindowsC.py

Content:

```
import psutil
import time

while True:
processes = []
for proc in psutil.process_iter(['pid', 'name', 'cpu_percent']):
try:
processes.append(proc.info)
except (psutil.NoSuchProcess, psutil.AccessDenied):
pass

processes = sorted(processes, key=lambda p: p['cpu_percent'], reverse=True)

print("\nTop 5 CPU-consuming processes:")
for proc in processes[:5]:
print(f"PID: {proc['pid']} | Name: {proc['name']} | CPU: {proc['cpu_percent']}%")

time.sleep(2)
```

---------- **FileName:** 15WindowsC.py

Content:

```
import psutil

name = input("Enter the process name (like explorer.exe): ")

found = []
for proc in psutil.process_iter(['pid', 'name']):
if proc.info['name'] and proc.info['name'].lower() == name.lower():
found.append(proc.info['pid'])

if found:
print(f"Yes, {name} is running with PID(s): {', '.join(map(str, found))}")
else:
print(f"No, I couldn’t find {name} running right now.")
```

---------- **FileName:** 16WindowsC.py

Content:

```
import subprocess

service_name = input("Enter the Windows service name (e.g., Spooler): ")

print(f"Attempting to restart the {service_name} service...")

# Stop the service
stop = subprocess.run(f'net stop {service_name}', capture_output=True, text=True, shell=True)
print(stop.stdout.strip())

# Start the service
start = subprocess.run(f'net start {service_name}', capture_output=True, text=True, shell=True)
print(start.stdout.strip())

print(f"{service_name} service restart attempt completed.")
```

---------- **FileName:** 17WindowsC.py

Content:

```
import os
import psutil
import subprocess

output = subprocess.run(["sc", "query", "type=", "service", "state=", "all"], capture_output=True,
text=True)
lines = output.stdout.splitlines()

services = []
service_name = ""
display_name = ""
status = ""

for line in lines:
line = line.strip()
if line.startswith("SERVICE_NAME:"):
service_name = line.split(":", 1)[1].strip()
elif line.startswith("DISPLAY_NAME:"):
display_name = line.split(":", 1)[1].strip()
elif line.startswith("STATE"):
status = "Running" if "4" in line else "Stopped"
services.append((display_name, status))

print("\nList of all services and their status:\n")
for display_name, status in services:
print(f"- {display_name} is currently {status}.")
```

```
print("you can type ]automatic Start] should you type like this if you want make it start automatic")
print("type exit to quit")
Input = input("type what is the proccess you want startup: ")

if "automatic Start" or "automatic start"in Input:
os.system(f"sc config {Input} start=auto")
if "exit" or "quit" in Input:
exit()
else:
print("Erorr")
os.system(f"net start {Input}")
```

FileName: 18WindowsC.py

Content:

```
import subprocess

def manage_service(service_name, action):
if action not in ["start", "stop", "restart"]:
print("Invalid action. Use start, stop, or restart.")
return

if action == "restart":
print(f"Restarting {service_name} service...")
subprocess.run(f'net stop {service_name}', shell=True)
subprocess.run(f'net start {service_name}', shell=True)
else:
print(f"{action.capitalize()}ing {service_name} service...")
subprocess.run(f'net {action} {service_name}', shell=True)

service = input("Enter the service name (e.g., Spooler): ")
choice = input("Do you want to start, stop, or restart it? ").lower()

manage_service(service, choice)
```

FileName: 19WindowsC.py

Content:

```
import psutil
import time
import os

def show_processes():
os.system("cls")
print(f"{'PID':<10}{'Name':<25}{'CPU %':<10}{'Memory %':<10}")
print("-" * 55)

for proc in psutil.process_iter(['pid', 'name', 'cpu_percent', 'memory_percent']):
```

```
try:
pid = proc.info['pid']
name = proc.info['name'][:23] if proc.info['name'] else "Unknown"
cpu = proc.info['cpu_percent']
mem = round(proc.info['memory_percent'], 2)
print(f"{pid:<10}{name:<25}{cpu:<10}{mem:<10}")
except (psutil.NoSuchProcess, psutil.AccessDenied):
continue

while True:
show_processes()
time.sleep(5)
```

FileName: 1WindowsC.py

Content:

```
import os

version = os.popen('ver').read().strip()
print(version)
```

FileName: 20WindowsC.py

Content:

```
import psutil
import getpass

username = input("Enter the username whose processes you want to terminate: ")

killed = []
failed = []

for proc in psutil.process_iter(['pid', 'name', 'username']):
try:
if proc.info['username'] and proc.info['username'].lower() == username.lower():
proc.kill()
killed.append((proc.info['pid'], proc.info['name']))
except (psutil.NoSuchProcess, psutil.AccessDenied):
failed.append((proc.info['pid'], proc.info['name']))

print("\n■ Scan complete.")
if killed:
print(f"■ Terminated {len(killed)} process(es) for user '{username}':")
for pid, name in killed:
print(f" • {name} (PID {pid})")
else:
print(f"■■ No processes found for user '{username}'.")

if failed:
```

```
print(f"\n■■ Could not terminate {len(failed)} process(es) due to permission issues.")
```

---------- **FileName:** 21WindowsC.py

Content:

```
import os

path = r"C:\Users\Public"

print(f"\n■ Listing contents of: {path}\n")

try:
for item in os.listdir(path):
full_path = os.path.join(path, item)
if os.path.isdir(full_path):
print(f"[DIR] {item}")
else:
print(f"[FILE] {item}")
except PermissionError:
print("■■ Access denied to some items.")
except FileNotFoundError:
print("■ The path does not exist.")
```

---------- **FileName:** 22WindowsC.py

Content:

```
import os


Input_fileToCp = input("enter the folder you want to cp: ")
Input_folderTp = input("enter the path you want to copy: ")

os.system("copy " + Input_fileToCp + Input_folderTp)
```

---------- **FileName:** 23WindowsC.py

Content:

```
import time
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

class Monitor(FileSystemEventHandler):
def on_created(self, event):
print(f"Created: {event.src_path}")
def on_deleted(self, event):
print(f"Deleted: {event.src_path}")
```

```
def on_modified(self, event):
if not event.is_directory:
print(f"Modified: {event.src_path}")
def on_moved(self, event):
print(f"Moved: {event.src_path} → {event.dest_path}")

def watch(path):
h = Monitor()
o = Observer()
o.schedule(h, path, recursive=True)
o.start()
print(f"Watching: {path}")
try:
while True:
time.sleep(1)
except KeyboardInterrupt:
o.stop()
o.join()

if __name__ == "__main__":
p = input("Enter directory path: ").strip()
watch(p)
```

---------- **FileName:** 24WindowsC.py

Content:

```
import os

path = input("Enter folder path: ").strip()
for f in os.listdir(path):
fp = os.path.join(path, f)
if os.path.isfile(fp):
print(f"{f} - {os.path.getsize(fp)} bytes")
```

---------- **FileName:** 25WindowsC.py

Content:

```
import os
import shutil
import schedule
import time
from datetime import datetime

SOURCE = input("Enter the folder you want to back up: ").strip()
DESTINATION = input("Enter the folder where backups should be saved: ").strip()

def backup():
today = datetime.now().strftime("%Y-%m-%d")
```

```
target = os.path.join(DESTINATION, f"backup-{today}")
if not os.path.exists(target):
shutil.copytree(SOURCE, target)
print(f"Backup created: {target}")
else:
print(f"Backup for {today} already exists.")

schedule.every().day.at("10:00").do(backup)

print("Backup scheduler started. Press Ctrl+C to stop.")
while True:
schedule.run_pending()
time.sleep(1)
```

FileName: 26WindowsC.py

Content:

```
import os

root = input("Enter folder path: ").strip()
name = input("Enter file name: ").strip()

for p, d, f in os.walk(root):
if name in f:
print(os.path.join(p, name))
```

FileName: 27Windows.py

Content:

```
import time
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

class Monitor(FileSystemEventHandler):
def on_created(self, event):
if not event.is_directory:
print(f"New file: {event.src_path}")

path = r"C:\Windows\Temp"
h = Monitor()
o = Observer()
o.schedule(h, path, recursive=False)
o.start()
print(f"Watching: {path}")
try:
while True:
time.sleep(1)
except KeyboardInterrupt:
```

```
o.stop()
o.join()
```

---------- **FileName:** 28WindowsC.py

Content:

```
import os



Input = input("what the file you want unzip it (should .zip)")


os.system("tar -xf " + Input)
```

---------- **FileName:** 29WindowsC.py

Content:

```
import os
import time

path = input("Enter folder path: ").strip()
now = time.time()
day = 24 * 60 * 60

for f in os.listdir(path):
fp = os.path.join(path, f)
if os.path.isfile(fp):
if now - os.path.getmtime(fp) <= day:
print(fp)
```

---------- **FileName:** 2WindowsC.py

Content:

```
import os

computer_name = os.environ.get("COMPUTERNAME")
user_name = os.getlogin()

print(f"Computer Name: {computer_name}")
print(f"Logged-in User: {user_name}")
```

---------- **FileName:** 30WindowsC.py

Content:

```
import os

path = input("Enter folder path: ").strip()
limit = 100 * 1024 * 1024

for p, d, f in os.walk(path):
for file in f:
fp = os.path.join(p, file)
if os.path.isfile(fp) and os.path.getsize(fp) > limit:
print(fp, "-", os.path.getsize(fp) // (1024 * 1024), "MB")
```

---------- **FileName:** 3WindowsC.py

Content:

```
import os

for key, value in os.environ.items():
print(f"{key} = {value}")
```

---------- **FileName:** 4WindowsC.py

Content:

```
import os

cwd = os.getcwd()
print(f"Current Working Directory: {cwd}")
```

---------- **FileName:** 5WindowsC.py

Content:

```
import platform
import subprocess

def get_windows_version():
try:
version = subprocess.check_output(
["wmic", "os", "get", "Caption"],
stderr=subprocess.DEVNULL,
stdin=subprocess.DEVNULL
).decode("utf-8", errors="ignore").split("\n")[1].strip()
if version:
return version
```

```
else:
return f"Windows {platform.release()}"
except Exception:
return f"Windows {platform.release()}"

def get_architecture():
return platform.architecture()[0]

print("Windows Version:", get_windows_version())
print("System Architecture:", get_architecture())
```

---------- **FileName:** 6WindowsC.py

Content:

```
import os



home_directory = os.path.expanduser("~")
print(home_directory)
```

---------- **FileName:** 7WindowsC.py

Content:

```
import os

if os.name == "nt":
print("This system is Windows")
else:
print("This system is NOT Windows")
```

---------- **FileName:** 8WindowsC.py

Content:

```
import psutil
from datetime import datetime

boot_timestamp = psutil.boot_time()
boot_time = datetime.fromtimestamp(boot_timestamp)

print("System boot time:", boot_time.strftime("%Y-%m-%d %H:%M:%S"))
```

---------- **FileName:** 9WindowsC.py

Content:
```
import platform
import psutil

# Get CPU name
cpu_name = platform.processor()

# Get number of physical cores
physical_cores = psutil.cpu_count(logical=False)

# Get number of logical cores (includes hyper-threading)
logical_cores = psutil.cpu_count(logical=True)

print(f"CPU Name: {cpu_name}")
print(f"Physical Cores: {physical_cores}")
print(f"Logical Cores: {logical_cores}")
```

--- **FileName:**
main.py

Content:
```
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Table, TableStyle
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib import colors
import os

def read_file_content(filename):
"""Safely read file content as text."""
try:
with open(filename, "r", encoding="utf-8") as f:
return f.read()
except Exception as e:
return f"Error reading file: {e}"

def add_file_section(story, styles, filename, content):
"""Add one file’s section to the PDF."""
section = f"""
```

FileName: {filename}

Content:
```
{content.replace('\n', '
')}

```

"""
story.append(Paragraph(section, styles["Normal"]))
story.append(Spacer(1, 20))

def make_pdf():
doc = SimpleDocTemplate("result.pdf", pagesize=A4)
```
```

```
styles = getSampleStyleSheet()
story = []

include_all = input("Do you want to include all files in this folder? (yes/no): ").strip().lower()

if include_all == "yes":
py_files = [f for f in os.listdir() if f.endswith(".py")]
txt_files = [f for f in os.listdir() if f.endswith(".txt")]

# ---- Add .py files sections ----
if py_files:
story.append(Paragraph("**Python Files**", styles["Heading2"]))
story.append(Spacer(1, 10))
for f in py_files:
content = read_file_content(f)
add_file_section(story, styles, f, content)

# ---- Add .txt files sections ----
if txt_files:
story.append(Paragraph("**Text Files**", styles["Heading2"]))
story.append(Spacer(1, 10))
for f in txt_files:
content = read_file_content(f)
add_file_section(story, styles, f, content)

# ---- Python Files Table ----
if py_files:
data = [["Python File Name", "Preview (first 80 chars)"]]
for f in py_files:
code = read_file_content(f)
preview = (code[:80] + "...") if len(code) > 80 else code
data.append([f, preview])
table = Table(data, colWidths=[200, 300])
table.setStyle(TableStyle([
("BACKGROUND", (0, 0), (-1, 0), colors.lightblue),
("GRID", (0, 0), (-1, -1), 1, colors.black),
("ALIGN", (0, 0), (-1, -1), "LEFT"),
]))
story.append(Paragraph("**Python Files Summary**", styles["Heading3"]))
story.append(table)
story.append(Spacer(1, 20))

# ---- Text Files Table ----
if txt_files:
data = [["Text File Name", "Preview (first 80 chars)"]]
for f in txt_files:
text = read_file_content(f)
preview = (text[:80] + "...") if len(text) > 80 else text
data.append([f, preview])
table = Table(data, colWidths=[200, 300])
table.setStyle(TableStyle([
("BACKGROUND", (0, 0), (-1, 0), colors.lightgreen),
("GRID", (0, 0), (-1, -1), 1, colors.black),
("ALIGN", (0, 0), (-1, -1), "LEFT"),
]))
story.append(Paragraph("**Text Files Summary**", styles["Heading3"]))
```

```
story.append(table)

print("■ Added all .py and .txt files to result.pdf")

else:
# ---- Single file mode ----
filename = input("Enter the file name (.py or .txt): ").strip()
if not os.path.exists(filename):
print("■ File not found.")
return
if not (filename.endswith(".py") or filename.endswith(".txt")):
print("■ Only .py and .txt files are allowed.")
return

content = read_file_content(filename)
add_file_section(story, styles, filename, content)
print(f"■ Added {filename} to result.pdf")

# ---- Build PDF ----
doc.build(story)
print("■ PDF created successfully: result.pdf")

if __name__ == "__main__":
make_pdf()
```

Python Files Summary

Python File Name	Preview (first 80 chars)
10WindowsC.py	import os var_name = os.getcwd() # the environment variable to check if var_n...
11WindowsC.py	import psutil for proc in psutil.process_iter(['pid', 'name']): print(f"PID...
12WindowsC.py	import psutil import os for proc in psutil.process_iter(['pid', 'name']): ...
13WindowsC.py	import os print("the code programming for Windows!!!") print("should the app in...

14WindowsC.py	import psutil
	import time
	
	while True:
	  processes = []
	  for proc in psutil...

15WindowsC.py	import psutil name = input("Enter the process name (like explorer.exe): ") fou...
16WindowsC.py	import subprocess service_name = input("Enter the Windows service name (e.g., S...
17WindowsC.py	import os import psutil import subprocess output = subprocess.run(["sc", "query...
18WindowsC.py	import subprocess def manage_service(service_name, action): if action not i...
19WindowsC.py	import psutil import time import os def show_processes(): os.system("cls") ...
1WindowsC.py	import os version = os.popen('ver').read().strip() print(version)
20WindowsC.py	import psutil import getpass username = input("Enter the username whose process...
21WindowsC.py	import os path = r"C:\Users\Public" print(f"\n■ Listing contents of: {path}\n"...
22WindowsC.py	import os Input_fileToCp = input("enter the folder you want to cp: ") Input_fo...

23WindowsC.py	import time from watchdog.observers import Observer from watchdog.events import ...
24WindowsC.py	import os path = input("Enter folder path: ").strip() for f in os.listdir(path)...
25WindowsC.py	import os import shutil import schedule import time from datetime import datetim...
26WindowsC.py	import os root = input("Enter folder path: ").strip() name = input("Enter file ...
27Windows.py	import time from watchdog.observers import Observer from watchdog.events import ...
28WindowsC.py	import os Input = input("what the file you want unzip it (should .zip)") o...
29WindowsC.py	import os import time path = input("Enter folder path: ").strip() now = time.ti...
2WindowsC.py	import os computer_name = os.environ.get("COMPUTERNAME") user_name = os.getlogi...
30WindowsC.py	import os path = input("Enter folder path: ").strip() limit = 100 * 1024 * 1024...
3WindowsC.py	import os for key, value in os.environ.items(): print(f"{key} = {value}")
4WindowsC.py	import os cwd = os.getcwd() print(f"Current Working Directory: {cwd}")

5WindowsC.py	import platform
	import subprocess
	
	def get_windows_version():
	  try:
	  v...

6WindowsC.py	import os home_directory = os.path.expanduser("~") print(home_directory)

7WindowsC.py	import os
	
	if os.name == "nt":
	  print("This system is Windows")
	else:
	  pri...

8WindowsC.py	import psutil from datetime import datetime boot_timestamp = psutil.boot_time()...

9WindowsC.py	import platform
	import psutil
	
	# Get CPU name
	cpu_name = platform.processor()
	
	#...

main.py	from reportlab.lib.pagesizes import A4 from reportlab.platypus import SimpleDocT...