1

```python
import sys
import os

def list_env_variables():
    for key in sorted(os.environ):
        print(f"{key}={os.environ[key]}")

def filter_env_variables(filters):
    for key in sorted(os.environ):
        if any(f in key for f in filters):
            print(f"{key}={os.environ[key]}")


if __name__ == "__main__":
    args = sys.argv[1:]

    if not args:
        list_env_variables()
    else:
        filter_env_variables(args)
```

```
AMENT_PREFIX_PATH=/home/kac/ros2_/install/zed_ros2:/home/kac/ros2_/install/zed_
rapper:/home/kac/ros2_/install/zed_components:/home/kac/ros2_/install/zed_inter
aces:/home/kac/ros2_/install/tools:/home/kac/ros2_/install/simulation_endpoint:
home/kac/ros2_/install/scorpio_rviz_plugins:/home/kac/ros2_/install/scorpio_mov
it:/home/kac/ros2_/install/image_processing:/home/kac/ros2_/install/control:/ho
e/kac/ros2_/install/scorpio_interfaces:/home/kac/ros2_/install/scorpio_hardware
/home/kac/ros2_/install/ros_tcp_endpoint:/home/kac/ros2_/install/realsense2_des
ription:/home/kac/ros2_/install/realsense2_camera:/home/kac/ros2_/install/reals
nse2_camera_msgs:/home/kac/ros2_/install/localization:/home/kac/ros2_/install/d
scription:/home/kac/ros2_/install/bringup:/home/kac/isaac_ros_ws/install/isaac_
```

```
kac@kla:~/studia/python_syf/jezyki-skryptowe/l4$ python3 z1.py USER
USER=kac
USERNAME=kac
```

2.

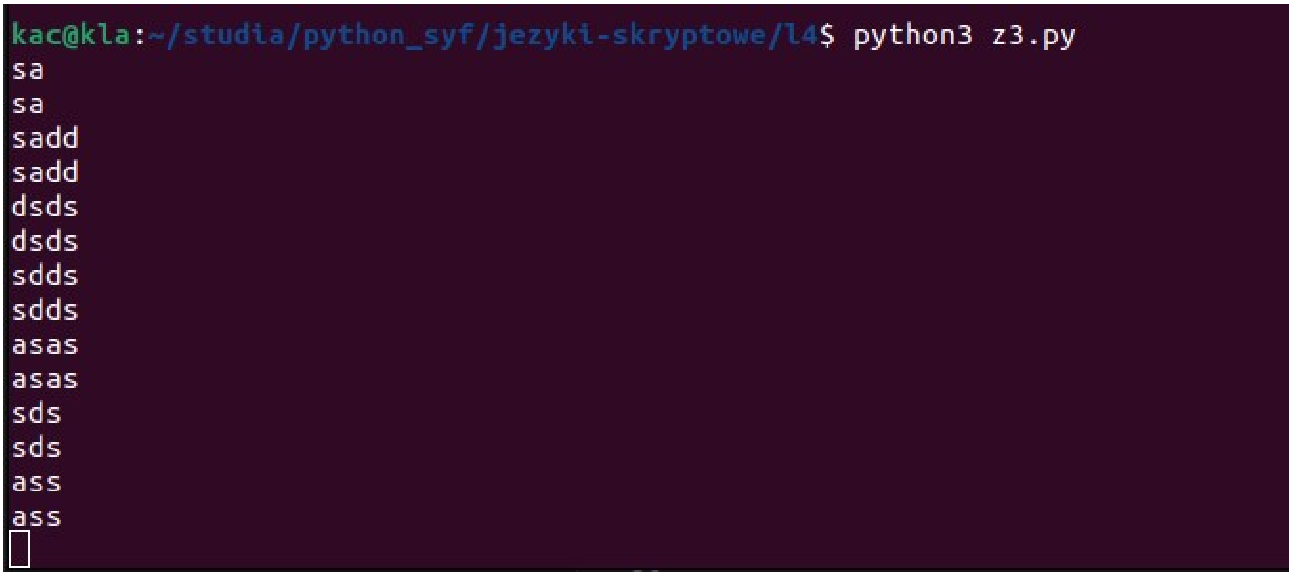
```python
import os
import sys

def list_path_directories():
    path_dirs = os.environ.get("PATH", "").split(os.pathsep)
    for directory in path_dirs:
        print(directory)

def list_executables():
    path_dirs = os.environ.get("PATH", "").split(os.pathsep)
    for directory in path_dirs:
        if os.path.isdir(directory):
            print(f"{directory}:")
            try:
                files = os.listdir(directory)
                executables = [f for f in files if os.path.isfile(os.path.join(directory, f)) and os.access(os.path.join(directory, f), os.X_OK)]
                for exe in executables:
                    print(f"\t{exe}")
            except:
                continue
            print()


if __name__ == "__main__":
    args = sys.argv[1:]

    if "--directories" in args:
        list_path_directories()
    if "--executables" in args:
        list_executables()
```

```
kac@kla:~/studia/python_syf/jezyki-skryptowe/l4$ python3 z2.py --directories
/usr/local/cuda/bin
/home/kac/isaac_ros_ws/install/isaac_ros_common/bin
/opt/ros/humble/bin
/usr/lib/jvm/java-11-openjdk-amd64/bin
/home/kac/.opam/default/bin
```

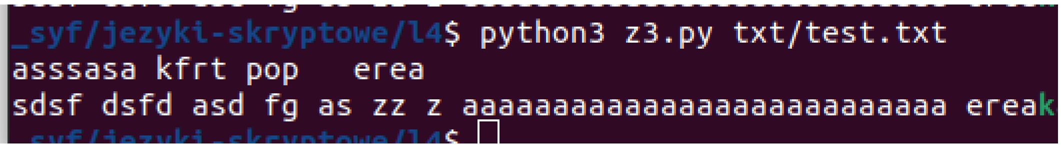
```
kac@kla:~/studia/python_syf/jezyki-skryptowe/l4$ python3 z2.py --executables
/usr/local/cuda/bin:
        cuda-gdb-python3.10-tui
        cuda-gdb-python3.12-tui
        nvprof
        nsight_ee_plugins_manage.sh
        ptxas
        cuda-gdb-minimal
        nsight-sys
        ncu-ui
```

3

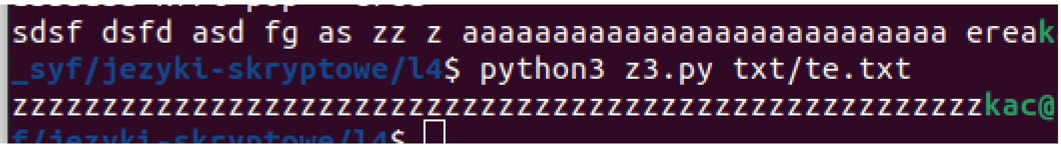
```python
import sys
import time

# Reading head of the file
def read_head(file, num_lines=10):
    try:
        with open(file, 'r', encoding='utf-8') as f:
            for _ in range(num_lines):
                line = f.readline()
                if not line:
                    break
                print(line, end='')
    except Exception as e:
        raise e

# Reading head of stdin
def read_head_stdin(num_lines=10):
    for _ in range(num_lines):
        line = sys.stdin.readline()
        if not line:
            break
        print(line, end='')

# Folowing file
def follow(file):
    try:
        with open(file, 'r', encoding='utf-8') as f:
            while True:
                line = f.readline()
                if line:
                    print(line, end='')
                else:
                    time.sleep(1)
    except Exception as e:
        raise e

def main():
    args = sys.argv[1:]
    num_lines = 10
    follow_mode = False
    file_path = None
    if (len(args)>3):
        raise Exception("Zbyt wiele argumentow")

    for arg in args:
        if arg.startswith('--lines='):
            try:
                num_lines = int(arg.split('=')[1])
            except Exception as e:
                raise Exception("Liczba nie jest intem w --lines")
        elif arg == '--follow':
            follow_mode = True
        else:
            file_path = arg

    if file_path:
        if follow_mode:
            follow(file_path)
        else:
            read_head(file_path, num_lines)
    else:
        read_head_stdin(num_lines)

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



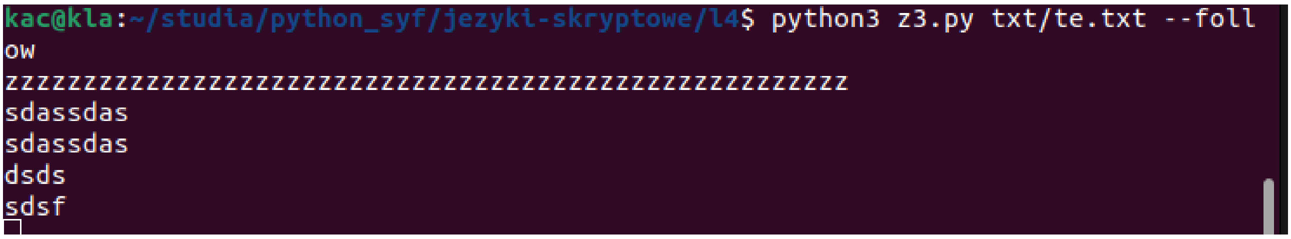
```
kac@kla:~/studia/python_syf/jezyki-skryptowe/l4$ python3 z3.py
sa
sa
sadd
sadd
dsds
dsds
sdds
sdds
asas
asas
sds
sds
ass
ass
```



```
_syf/jezyki-skryptowe/l4$ python3 z3.py txt/test.txt
asssasa kfrt pop    erea
sdsf dsfd asd fg as zz z aaaaaaaaaaaaaaaaaaaaaaaaaaa ereak
_syf/jezyki-skryptowe/l4$
```



```
sdsf dsfd asd fg as zz z aaaaaaaaaaaaaaaaaaaaaaaaaaa ereak
_syf/jezyki-skryptowe/l4$ python3 z3.py txt/te.txt
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzkac@
f/jezyki-skryptowe/l4$
```



```
kac@kla:~/studia/python_syf/jezyki-skryptowe/l4$ python3 z3.py txt/te.txt --foll
ow
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
sdassdas
sdassdas
dsds
sdsf
```

4

```python
import sys
import csv
from collections import Counter
import json

def analyze_file(file_path, with_counter=False):
    try:
        with open(file_path, 'r', encoding='utf-8') as f:
            content = f.read()
    except:
        raise Exception("File not found")

    char_count = len(content)
    words = content.split()
    word_count = len(words)
    lines = content.splitlines()
    line_count = len(lines)

    # Using counter for counting most commons
    char_counter = Counter(content)
    most_common_char = char_counter.most_common(1)[0][0] if char_counter else ''

    word_counter = Counter(words)
    most_common_word = word_counter.most_common(1)[0][0] if word_counter else ''

    # Writing output
    if with_counter:
        result = [file_path, char_count, word_count, line_count, json.dumps(char_counter),json.dumps(word_counter)]
    else:
        result = [file_path, char_count, word_count, line_count, most_common_char, most_common_word]

    writer = csv.writer(sys.stdout)
    writer.writerow(result)


if __name__ == "__main__":
    file_path = sys.stdin.readline().strip()
    if not file_path or file_path=='':
        raise Exception("No file_path given")
    # Added flag for 4.2
    if len(sys.argv) ==2 and sys.argv[1] == '--with_counters':
        analyze_file(file_path, True)
    else:
        analyze_file(file_path)
```

```
kac@kla:~/studia/python_syf/jezyki-skryptowe/l4$ python3 z4.py
txt/test.txt
txt/test.txt,81,13,2,a,erea
```

```
asssasa kfrt pop   erea
sdsf dsfd asd fg as zz z aaaaaaaaaaaaaaaaaaaaaaaaaa erea
```

4.2

```python
def run_analysis_on_file(filepath):
    )
    if result.returncode != 0:
        return None

    reader = csv.reader(result.stdout.strip().split('\n'))
    row = next(reader)
    return {
        "path": row[0],
        "chars": int(row[1]),
        "words": int(row[2]),
        "lines": int(row[3]),
        "most_common_chars_counter": json.loads(row[4]),
        "most_common_words_counter": json.loads(row[5]),
    }

def main(directory):
    # Analizing every file in directory
    results = []
    for file_name in os.listdir(directory):
        full_path = os.path.join(directory, file_name)
        if os.path.isfile(full_path):
            result = run_analysis_on_file(full_path)
            if result:
                results.append(result)

    if not results:
        print("Brak wyników.")
        return
    # Standard adding
    total_files = len(results)
    total_chars = sum(r["chars"] for r in results)
    total_words = sum(r["words"] for r in results)
    total_lines = sum(r["lines"] for r in results)
    # Adding from different files counters so it will return most commons from across files
    all_chars = Counter()
    all_words = Counter()
    for r in results:
        all_chars.update(r["most_common_char"])
        all_words.update(r["most_common_word"])
    most_common_char = all_chars.most_common(1)[0][0]
    most_common_word = all_words.most_common(1)[0][0]
    # Printing every data
    print("LICZBA PLIKÓW:", total_files)
    print("SUMA ZNAKÓW:", total_chars)
    print("SUMA SŁÓW:", total_words)
    print("SUMA WIERSZY:", total_lines)
    print("NAJCZĘSTSZY ZNAK:", most_common_char)
    print("NAJCZĘSTSZE SŁOWO:", most_common_word)

if __name__ == "__main__":
    if len(sys.argv) < 2:
        raise Exception ('Directory not given')
    else:
        main(sys.argv[1])
```

```
kac@kla:~/studia/python_syf/jezyki-skryptowe/l4$ python3 z4_2.py txt/
LICZBA_PLIKÓW: 2
SUMA_ZNAKÓW: 164
SUMA_SŁÓW: 18
SUMA_WIERSZY: 7
NAJCZĘSTSZY_ZNAK: z
NAJCZĘSTSZE_SŁOWO: sdassdas
```

5

mediaconvert.py > ...

```python
def log_to_json(history_folder, og_path, output_format, output_path, program):
    '''

    with open(destination, "w") as json_file:
        dump(info, json_file, indent=4)

    print(f'Log dumped to {destination}')

if __name__ == '__main__' and len(argv) >= 2:
    # Getting all files from directory
    folder = argv[1]
    files = [path.join(folder, file) for file in listdir(folder)]
    # Getting types of files from arguments
    target_format_for_audio = next((f.split('=')[-1] for f in argv if f.startswith('--audio')), (print('Using mp4') or 'mp4'))
    target_format_for_image = next((f.split('=')[-1] for f in argv if f.startswith('--image')), (print('Using png') or 'png'))
    # Making output folder
    output_folder = getenv("CONVERTED_DIR", 'converted')
    makedirs(output_folder, exist_ok=True)

    for filename in files:
        used_program = ''
        # Creating new file name
        filename_short = filename.split('/')[-1]
        new_filename = filename_short.split('.')[0]
        new_filename =  f'{datetime.now().strftime("%Y%m%d")}-{new_filename}'
        # Checking file type
        mimestart = mimetypes.guess_type(filename_short)[0]
        if mimestart != None:
            mimestart = mimestart.split('/')[0]
        else:
            print(f"Ignorowanie ścieżki: {filename}")
            continue
        # Using ffmpeg for audio and video
        if mimestart in ['audio','video']:
            new_filename = path.join(output_folder, new_filename+f'.{target_format_for_audio}')

            result = run(
                ['ffmpeg', '-y', '-i', filename, new_filename],
                capture_output=True,
                text=True,
            )
            used_program = 'ffmpeg'
            target_format = target_format_for_audio
        # Using imagemagick for
        elif mimestart == 'image':
            new_filename = path.join(output_folder, new_filename+f'.{target_format_for_image}')

            result = run([
                "convert",  filename, new_filename ],
                 capture_output=True, text=True)
            used_program = 'imagemagick'
            target_format = target_format_for_image
        else:
            print(f'Ścieżka nie jest audio: {filename}')
        # Making log
        if result.returncode == 0:
            log_to_json(
                    history_folder = output_folder,
                    og_path = filename,
                    output_format = target_format,
                    output_path = new_filename,
                    program = used_program
            )
        else:
            print(f'Conversion failed for: {filename}')
```

```
    },
    {
        "time": "2025-04-06 21:01:42.123020",
        "original path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/media/image.png",
        "output format": "png",
        "output path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/converted/20250406-image.png",
        "used program": "imagemagick"
    },
    {
        "time": "2025-04-06 21:02:24.375531",
        "original path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/media/01 - Alice In Chains - Them Bones.flac",
        "output format": "mp3",
        "output path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/converted/20250406-01 - Alice In Chains - Them Bones.mp3",
        "used program": "ffmpeg"
    },
    {
        "time": "2025-04-06 21:02:24.414231",
        "original path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/media/image.png",
        "output format": "jpg",
        "output path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/converted/20250406-image.jpg",
        "used program": "imagemagick"
    },
    {
        "time": "2025-04-06 21:02:46.020149",
        "original path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/media/image.png",
        "output format": "--image",
        "output path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/converted/20250406-image.--image",
        "used program": "imagemagick"
    },
    {
        "time": "2025-04-06 21:02:58.028955",
        "original path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/media/image.png",
        "output format": "jpg",
        "output path": "/home/kac/studia/python_syf/jezyki-skryptowe/l4/converted/20250406-image.jpg",
        "used program": "imagemagick"
    }
]
```


Do you want to install the recom