Code projects (python & simple)

- map the internet based on if i get a ping
  - map maker v3.py (functioning?) last ping 1.0.212.120 white
- sort imgs.py
- nzk-install
- nzk-help

map the internet based on if i get a ping

map the internet based on if i get a ping

map maker v3.py (functioning?) last ping 1.0.212.120 white

```
import os
import socket
import struct
from ping3 import ping
from PIL import Image

print("start running")



def ip_to_int(ip):
    int_ip = struct.unpack("!I", socket.inet_aton(ip))[0]
    return int_ip
def int_to_ip(i):
    ip = socket.inet_ntoa(struct.pack("!I", i))
    return ip
def hilbert_curve(n):
    points = [(0, 0)]
    for i in range(n):
        gray = [((k >> i) ^ (k >> (i + 1))) & 1 for k in range(2 ** i)]
        points = _rot(points, gray)
    return points

def _rot(points, gray):
    rot = [(1, 0), (0, 1), (-1, 0), (0, -1)]
    last = points[-1]
    for code in gray:
        last = (last[0] + rot[code][0], last[1] + rot[code][1])
        points.append(last)
    return points

def ping_ip(ip):
    try:
        response_time = ping(ip, timeout=1)
        return 1 if response_time is not None else 0
    except OSError:
        return -1

def main():
```

```
img_size = 65536
n = 16
print(f"img size ={img_size}, n={n} Creating a blank image with a white background")

img = Image.new("1", (img_size, img_size), color="grey")
pixels = img.load()
curve_filename = "hilbert_curve v2.txt"
if os.path.exists(curve_filename):
    with open(curve_filename, "r") as f:
        print("reading Hilbert curve coordinates file")
        curve_points = [tuple(map(int, line.strip().split())) for line in f.readlines()]
else:
    print("Generating Hilbert curve coordinates file")
    curve_points = hilbert_curve(n)
    with open(curve_filename, "w") as f:
        for x, y in curve_points:
            f.write(f"{x} {y}\n")

ip_range = 2 ** 32
start_ip_int = 0
if os.path.exists('ping_status.txt'):
    with open('ping_status.txt', 'r') as f:
        lines = f.readlines()
        if len(lines) > 0:
            last_line = lines[-1].strip()
            last_ip, last_result = last_line.split()
            start_ip_int = ip_to_int(last_ip) + 1
            print(f"Resuming from IP address {last_ip}, result = {last_result}")
with open('ping_status.txt', 'a') as f:
    print("pinging and writing to ping status.txt")
    for i in range(start_ip_int, ip_range):
        ip = int_to_ip(i)
        x, y = curve_points[i % len(curve_points)]
        result = ping_ip(ip)
        print(f"result {result} for ip {ip}")
        if result == -1:
            f.write(f"{ip} black\n")
            pixels[x, y] = 0
        else:
            pixels[x, y] = result
            if result == 0:
                f.write(f"{ip} white\n")
            else:
                f.write(f"{ip} black\n")
        if (i + 1) % (2 ** 24) == 0:
            img.save("ping_map.png")
```

```
    img.save("ping_map.png")
    print("Finished pinging IP addresses and saved final image")
if __name__ == '__main__':
    main()
```

sort imgs.py

`edit the code for the folder`

```
from PIL import Image
import os

unique_images = {}

# edit this
directory = r'J:\New folder'



for filename in os.listdir(directory):
    if filename.endswith('.png'):

        img = Image.open(os.path.join(directory, filename))

        if str(img.tobytes()) in unique_images:

            os.remove(os.path.join(directory, filename))
            print(f"Deleted {filename}")
        else:

            unique_images[str(img.tobytes())] = filename
            print(f"Kept {filename}")
```

this one will ask where you want to check

```
from PIL import Image
import os
unique_images = {}
directory = input("Enter the directory containing the PNG files: ")
print(f"Directory: {directory}")
for filename in os.listdir(directory):
    if filename.endswith('.png'):
        img = Image.open(os.path.join(directory, filename))
        if str(img.tobytes()) in unique_images:
```

```
            os.remove(os.path.join(directory, filename))
            print(f"Deleted {filename}")
        else:

            unique_images[str(img.tobytes())] = filename
            print(f"Kept {filename}")
```

nzk-install

```
#!/bin/bash

# Script to install an AppImage to /nzk/appImages and add it to the application menu
# update nzk-install with nkz-install --update
APPIMAGE_PATH=""
APP_NAME=""
OVERRIDE=false
UPDATE=false
HELP=false
HELPFILE=false
HELPFILE_PATH=""
SUDO=false
AUR_URL=""
GIT_DIR="/nzk/git"

while [[ $# -gt 0 ]]; do
    case $1 in
        --name=*|-n=*)
            APP_NAME="${1#*=}"
            shift
            ;;
        --override=*|-o=*)
            OVERRIDE_VALUE="${1#*=}"
            if [ "$OVERRIDE_VALUE" = "y" ] || [ "$OVERRIDE_VALUE" = "yes" ]; then
                OVERRIDE=true
            fi
            shift
            ;;
        --update)
            UPDATE=true
            shift
            ;;
        --help)
            HELP=true
            shift
```

```
            ;;
        --helpfile)
            HELPFILE=true
            shift
            ;;
        --helpfile=*)
            HELPFILE_PATH="${1#*=}"
            HELPFILE=true
            shift
            ;;
        --sudo)
            SUDO=true
            shift
            ;;
        --aur=*)
            AUR_URL="${1#*=}"
            shift
            ;;
        *)
            if [ -z "$APPIMAGE_PATH" ]; then
                APPIMAGE_PATH="$1"
            else
                echo "Unknown argument: $1"
                exit 1
            fi
            shift
            ;;
    esac
done

if [ "$HELP" = true ]; then
    echo "Usage: $0 <path to AppImage or executable> [--name=\"App Name\" | -n=\"App Name\"]
[--override=y | -o=y] [--update] [--help] [--helpfile | --helpfile=/path/to/helpfile] [--sudo]
[--aur=<url>]"
    echo "Options:"
    echo "  --name, -n: Set custom name for the app"
    echo "  --override, -o: Override existing alias without prompting"
    echo "  --update: Update the script itself"
    echo "  --help: Show this help"
    echo "  --helpfile: Create a default help file for the app"
```

```
    echo "  --helpfile=/path: Copy help file from the specified path"
    echo "  --sudo: Run the app with sudo"
    echo "  --aur=<url>: Install from AUR git repository"
    echo "Note: FUSE installation is only performed for AppImages."
    exit 0
fi

if [ "$UPDATE" = true ]; then
    echo "Updating nzk-install..."
    sudo cp "/home/naruzkurai/Coding/AppImage-to-start-link/nzk-install" "/usr/local/bin/nzk-
install"
    echo "nzk-install updated."
    exit 0
fi

if [ -n "$AUR_URL" ]; then
    # Handle AUR install
    if [ -f /etc/os-release ]; then
        . /etc/os-release
        if [ "$ID" != "arch" ] && [ "$ID" != "manjaro" ] && [ "$ID" != "cachyos" ] && [ "$ID"
!= "endeavouros" ]; then
            echo "AUR install is only supported on Arch-based systems."
            exit 1
        fi
    else
        echo "Cannot detect OS. AUR install may not work."
    fi
    if ! command -v git >/dev/null 2>&1; then
        echo "Git is not installed. Please install git first."
        exit 1
    fi
    sudo mkdir -p "$GIT_DIR"
    sudo chown -R $(whoami):$(whoami) "$GIT_DIR"
    REPO_NAME=$(basename "$AUR_URL" .git)
    CLONE_DIR="$GIT_DIR/$REPO_NAME"
    if [ -d "$CLONE_DIR" ]; then
        echo "Directory $CLONE_DIR already exists. Updating..."
        cd "$CLONE_DIR" || exit 1
        git pull
    else
```

```
        git clone "$AUR_URL" "$CLONE_DIR"
        cd "$CLONE_DIR" || exit 1
    fi
    # Build and install
    makepkg -si
    echo "Package installed from $AUR_URL"
    exit 0
fi

if [ -z "$APPIMAGE_PATH" ] && [ "$HELPFILE" = false ] && [ -z "$AUR_URL" ]; then
    echo "Usage: $0 <path to AppImage or executable> [--name=\"App Name\" | -n=\"App Name\"]
[--override=y | -o=y] [--update] [--help] [--helpfile | --helpfile=/path/to/helpfile] [--sudo]
[--aur=<url>]"
    exit 1
fi

if [ ! -f "$APPIMAGE_PATH" ]; then
    echo "Error: File not found at $APPIMAGE_PATH"
    exit 1
fi

# Handle tar.gz extraction
if [[ "$APPIMAGE_PATH" == *.tar.gz ]]; then
    EXTRACT_DIR="/tmp/$(basename "$APPIMAGE_PATH" .tar.gz)"
    rm -rf "$EXTRACT_DIR"
    mkdir -p "$EXTRACT_DIR"
    tar -xzf "$APPIMAGE_PATH" -C "$EXTRACT_DIR" || { echo "Failed to extract $APPIMAGE_PATH";
exit 1; }
    # Check if extracted to a single directory
    CONTENTS=("$EXTRACT_DIR"/*)
    if [ ${#CONTENTS[@]} -eq 1 ] && [ -d "${CONTENTS[0]}" ]; then
        APPIMAGE_PATH="${CONTENTS[0]}"
    else
        # Find the main executable
        EXECUTABLE=$(find "$EXTRACT_DIR" -type f -executable | head -1)
        if [ -z "$EXECUTABLE" ]; then
            echo "No executable found in extracted $APPIMAGE_PATH"
            exit 1
        fi
        APPIMAGE_PATH="$EXECUTABLE"
```

```
    fi
fi

# Detect if it's an AppImage
IS_APPIMAGE=false
if [[ "$APPIMAGE_PATH" == *.AppImage ]]; then
    IS_APPIMAGE=true
fi

DEST_DIR="/nzk-bin"

# Create the directory if it doesn't exist
sudo mkdir -p "$DEST_DIR"

# Set permissions to allow writing and executing
sudo chmod 755 "$DEST_DIR"

# Copy the AppImage or dir
if [ -d "$APPIMAGE_PATH" ]; then
    # Directory install
    EXECUTABLE_NAME=$(basename "$APPIMAGE_PATH")
    sudo cp -r "$APPIMAGE_PATH" "$DEST_DIR/"
    EXECUTABLE_REL="./$(find "$APPIMAGE_PATH" -type f -executable | head -1 | sed
"s|^$APPIMAGE_PATH/||")"
else
    # File install
    EXECUTABLE_NAME=$(basename "$APPIMAGE_PATH")
    sudo cp "$APPIMAGE_PATH" "$DEST_DIR/$EXECUTABLE_NAME"
    EXECUTABLE_REL="./$EXECUTABLE_NAME"
fi

# Make it executable
if [ -f "$DEST_DIR/$EXECUTABLE_NAME" ]; then
    sudo chmod +x "$DEST_DIR/$EXECUTABLE_NAME"
fi

install_fuse() {
    if [ -f /etc/os-release ]; then
        . /etc/os-release
        case $ID in
```

```
            arch|manjaro|cachyos|endeavouros)
                if ! pacman -Q fuse2 > /dev/null 2>&1; then
                    echo "Detected Arch-based system. Installing fuse2..."
                    sudo pacman -S --noconfirm fuse2
                fi
                ;;
            ubuntu|debian|pop|elementary|zorin|linuxmint)
                if ! dpkg -l libfuse2 | grep -q ^ii; then
                    echo "Detected Debian/Ubuntu-based system. Installing libfuse2..."
                    sudo apt update && sudo apt install -y libfuse2
                fi
                ;;
            fedora|rhel|centos|rocky|almalinux)
                if ! rpm -q fuse-libs > /dev/null 2>&1; then
                    echo "Detected Fedora/RHEL-based system. Installing fuse-libs..."
                    sudo dnf install -y fuse-libs
                fi
                ;;
            opensuse*|sles)
                if ! rpm -q fuse > /dev/null 2>&1; then
                    echo "Detected openSUSE. Installing fuse..."
                    sudo zypper install -y fuse
                fi
                ;;
            *)
                echo "Unknown OS: $ID. Please install FUSE manually (libfuse2, fuse2, or
equivalent)."
                ;;
        esac
    else
        echo "Cannot detect OS. Please install FUSE manually."
    fi
}

# Install FUSE if needed for AppImage to run
if [ "$IS_APPIMAGE" = true ]; then
    install_fuse
fi

# Set default name if not provided
```

```
if [ -z "$APP_NAME" ]; then
    APP_NAME="${EXECUTABLE_NAME%.AppImage}"
fi

# Create command name (replace spaces with dashes)
COMMAND_NAME=$(echo "$APP_NAME" | tr ' ' '-')
ALIAS_PATH="/usr/local/bin/nzk-$COMMAND_NAME"

# Check if alias already exists
if [ -e "$ALIAS_PATH" ]; then
    if [ "$OVERRIDE" = false ]; then
        echo "nzk-$COMMAND_NAME already exists. Override? (y/n)"
        read -r answer
        if [ "$answer" != "y" ] && [ "$answer" != "yes" ]; then
            echo "Aborting."
            exit 1
        fi
    fi
    sudo rm -f "$ALIAS_PATH"
fi

# Create alias (script for dir or symlink for file)
if [ -d "$APPIMAGE_PATH" ]; then
    # For directory
    if [ "$SUDO" = true ]; then
        sudo tee "$ALIAS_PATH" > /dev/null << EOF
#!/bin/bash
cd "$DEST_DIR/$EXECUTABLE_NAME"
sudo "$EXECUTABLE_REL" "\$@"
EOF
    else
        sudo tee "$ALIAS_PATH" > /dev/null << EOF
#!/bin/bash
cd "$DEST_DIR/$EXECUTABLE_NAME"
"$EXECUTABLE_REL" "\$@"
EOF
    fi
else
    # For file
    if [ "$SUDO" = true ]; then
```

```
        sudo tee "$ALIAS_PATH" > /dev/null << EOF
#!/bin/bash
sudo "$DEST_DIR/$EXECUTABLE_NAME" "\$@"
EOF
    else
        sudo ln -s "$DEST_DIR/$EXECUTABLE_NAME" "$ALIAS_PATH"
    fi
fi
sudo chmod +x "$ALIAS_PATH"

# Create help file if requested
if [ "$HELPFILE" = true ]; then
    HELP_DIR="/bin/nzk-apps/helpfiles/$COMMAND_NAME"
    sudo mkdir -p "$HELP_DIR"
    HELP_FILE="$HELP_DIR/helpfile"
    if [ -n "$HELPFILE_PATH" ]; then
        if [ -f "$HELPFILE_PATH" ]; then
            sudo cp "$HELPFILE_PATH" "$HELP_FILE"
            echo "Help file copied from $HELPFILE_PATH to $HELP_FILE"
        else
            echo "Error: Help file not found at $HELPFILE_PATH"
            exit 1
        fi
    else
        FILE_TYPE="Executable"
        if [ "$IS_APPIMAGE" = true ]; then
            FILE_TYPE="AppImage"
        fi
        sudo tee "$HELP_FILE" > /dev/null << EOF
Help for $APP_NAME

$FILE_TYPE: $EXECUTABLE_NAME
Location: $DEST_DIR/$EXECUTABLE_NAME
Alias: $ALIAS_PATH

To run the app:
- Use the alias: nzk-$COMMAND_NAME
- Or run directly: $DEST_DIR/$EXECUTABLE_NAME
- Or from the application menu.
```

```
For more information, check the $FILE_TYPE documentation.
EOF
        echo "Help file created at $HELP_FILE"
    fi
fi

# Create desktop file
DESKTOP_DIR="$HOME/.local/share/applications"
mkdir -p "$DESKTOP_DIR"

DESKTOP_FILE="$DESKTOP_DIR/$COMMAND_NAME.desktop"

cat > "$DESKTOP_FILE" << EOF
[Desktop Entry]
Version=1.0
Type=Application
Name=$APP_NAME
Exec=$ALIAS_PATH
Icon=application-x-executable
Terminal=false
Categories=Utility;
EOF

chmod +x "$DESKTOP_FILE"

echo "$FILE_TYPE installed to $DEST_DIR/$EXECUTABLE_NAME"
echo "Desktop entry created at $DESKTOP_FILE"
echo "Alias created: nzk-$COMMAND_NAME"
```

nzk-help

```
#!/bin/bash

# nzk-help: Display help for nzk-installed apps

if [ $# -eq 0 ]; then
    echo "Usage: nzk-help <appname>"
    echo "Available apps with help files:"
    if [ -d /bin/nzk-apps/helpfiles ]; then
        ls /bin/nzk-apps/helpfiles
    else
        echo "No help files found."
    fi
    exit 1
fi

APP_NAME="$1"
HELP_FILE="/bin/nzk-apps/helpfiles/$APP_NAME/helpfile"

if [ -f "$HELP_FILE" ]; then
    cat "$HELP_FILE"
else
    echo "No help file found for '$APP_NAME'."
    echo "Available apps:"
    if [ -d /bin/nzk-apps/helpfiles ]; then
        ls /bin/nzk-apps/helpfiles
    fi
    exit 1
fi
```