Работа с папками в Python: полное руководство

Работа с папками в Python является одним из фундаментальных навыков для любого разработчика. Python предоставляет мощные встроенные модули os и shutil, которые позволяют выполнять все основные операции с файловой системой: создание, копирование, перемещение, переименование и удаление папок.

Получение и изменение рабочего каталога

Перед началом работы с папками важно знать, в какой директории вы находитесь, и уметь изменять текущий рабочий каталог. При указании путей внутри приложения НИКОГДА не используются абсолютные пути, поскольку данная программа в таком случае становится не переносимой на другую систему без изменения исходного кода программы.

Получение текущего рабочего каталога:

```
import os

current_directory = os.getcwd()
print(f"Текущий рабочий каталог: {current_directory}")
```

Изменение текущего рабочего каталога:

```
import os

new_directory = "/path/to/new/directory"
os.chdir(new_directory)
print(f"Текущий рабочий каталог изменен на: {os.getcwd()}")
```

Создание новых папок

Python предлагает два основных способа создания папок: mkdir() для создания одной папки и makedirs() для создания вложенной структуры каталогов.

```
import os

# Создание новой папки
os.mkdir('новая_папка')

# Создание новой и промежуточной папки
os.makedirs('промежуточная_папка/новая_папка', exist_ok=True)
# exist_ok=True позволяет при наличии такой папки не вызывать ошибку
```

Важно: Параметр exist_ok=True предотвращает возникновение ошибки FileExistsError, если папка уже существует.

Проверка существования папки

Перед созданием или удалением папки рекомендуется проверить её существование:

```
import os

folder_path = 'моя_папка'
if os.path.exists(folder_path):
    print(f"Папка {folder_path} уже существует")
else:
    os.mkdir(folder_path)
    print(f"Папка {folder_path} создана")
```

Копирование папок

Для копирования папок со всем содержимым используется функция copytree() из модуля shutil:

```
import shutil

# Копирование папки со всем содержимым
shutil.copytree('исходная_папка', 'новая_папка')
```

Примечание: Целевая папка не должна существовать заранее, иначе возникнет ошибка. Лучшей практикой будет не использовать данную команду без проверки существования папки, вообще лучше не использовать команды без первичной проверки, таким образом мы защищаем себя от обработки типовых исключений и упрощаем код.

Хороший код — это не тот, что умеет падать и подниматься за счёт исключений, а тот, что изначально построен так, чтобы исключительные ситуации не возникали по логике работы. Лучшая практика — не использовать команды, которые могут привести к ошибке, без предварительных проверок. Мы не лечим симптомы (через try/except), а предотвращаем их причину логикой программы.

Перемещение и переименование папок

Для перемещения и переименования папок используются функции rename() из модуля os и move() из модуля shutil:

```
import os
import shutil

# Переименование папки (в той же директории)
os.rename('старое_имя', 'новое_имя')

# Перемещение папки в другую директорию
shutil.move('исходная_папка', 'целевая_папка/исходная_папка')

# Перемещение с переименованием
shutil.move('исходная_папка', 'целевая_папка/новое_имя')
```

Удаление папок

Python предоставляет несколько способов удаления папок в зависимости от того, пустая папка или содержит файлы:

```
import os
import shutil

# Удаление пустой папки
os.rmdir('пустая_папка')

# Удаление папки и всего её содержимого
shutil.rmtree('папка_с_содержимым')
```

Осторожно: Функция rmtree() удаляет папку безвозвратно вместе со всем содержимым! Применения с пользовательскими файлами без запроса на удаление запрещено!

Получение списка файлов и папок

Для просмотра содержимого директории используйте следующие методы:

```python
import os

# Получение списка всех элементов в директории
items = os.listdir('.')
print("Содержимое текущей директории:", items)

# Получение только папок
folders = [item for item in os.listdir('.') if os.path.isdir(item)]
print("Только папки:", folders)

# Получение только файлов
files = [item for item in os.listdir('.') if os.path.isfile(item)]
print("Только файлы:", files)
```

Практический пример: комплексная работа с папками

Рассмотрим полный пример, демонстрирующий основные операции с папками:

```python
import os
import shutil

# Создание новой директории
project_directory = "проект_папки"
os.makedirs(project_directory, exist_ok=True)

# Создание подпапок
subdirectories = ["исходники", "документы", "тесты"]
for subdir in subdirectories:
    os.makedirs(os.path.join(project_directory, subdir), exist_ok=True)

# Создание файла в одной из подпапок
file_path = os.path.join(project_directory, "исходники", "main.py")
with open(file_path, 'w', encoding='utf-8') as f:
    f.write("print('Hello, World!')")

# Копирование файла
copied_file_path = os.path.join(project_directory, "исходники",
"main_copy.py")
shutil.copy(file_path, copied_file_path)

# Создание архивной папки и перемещение файла
archive_dir = os.path.join(project_directory, "архив")
os.makedirs(archive_dir, exist_ok=True)
shutil.move(copied_file_path, os.path.join(archive_dir, "main_archived.py"))

# Вывод структуры проекта
print(f"Создана структура проекта в: {project_directory}")
for root, dirs, files in os.walk(project_directory):
    level = root.replace(project_directory, '').count(os.sep)
    indent = ' ' * 2 * level
    print(f"{indent}{os.path.basename(root)}/")
    subindent = ' ' * 2 * (level + 1)
    for file in files:
        print(f"{subindent}{file}")
```

Обработка ошибок при работе с папками

При работе с файловой системой всегда следует предусматривать обработку возможных ошибок:

```
import os
import shutil

try:
    # Попытка создания папки
    os.mkdir('новая_папка')
    print("Папка создана успешно")
except FileExistsError:
    print("Папка уже существует")
except PermissionError:
    print("Недостаточно прав для создания папки")
except OSError as e:
    print(f"Ошибка операционной системы: {e}")

try:
    # Попытка удаления папки
    shutil.rmtree('папка_для_удаления')
    print("Папка удалена успешно")
except FileNotFoundError:
    print("Папка не найдена")
except PermissionError:
    print("Недостаточно прав для удаления папки")
```

Заключение

Работа с папками в Python с использованием модулей os и shutil предоставляет все необходимые инструменты для эффективного управления файловой системой. Освоив эти основные операции, вы сможете создавать более сложные программы для автоматизации работы с файлами и директориями. Не забывайте всегда обрабатывать возможные исключения и проверять существование папок перед выполнением операций.