**Python XIII:**
**Работа с файлами**

Работа с файлами в Python выполняется с помощью встроенных функций и методов для работы с файловой системой. В общем случае, для работы с файлами в Python требуется выполнить три шага:

1. Открыть файл.
2. Выполнить необходимые операции с файлом (например, чтение или запись).
3. Закрыть файл.

Для открытия файла в Python используется функция open(). Эта функция возвращает объект файлового дескриптора, который можно использовать для выполнения операций чтения и записи. Например, чтобы открыть файл для чтения, можно использовать следующий код:

```
file = open('file.txt', 'r')
```

В этом примере открывается файл file.txt для чтения. Второй аргумент функции open() указывает режим открытия файла. Существуют различные режимы открытия файла, включая чтение ('r'), запись ('w') и добавление ('a'). Если режим не указан, используется режим чтения по умолчанию ('r').

После открытия файла можно выполнять операции чтения или записи. Например, чтобы прочитать содержимое файла, можно использовать метод read() объекта файлового дескриптора:

```
content = file.read()
```

После выполнения операций чтения или записи файл следует закрыть с помощью метода close():

```
file.close()
```

Не закрывать файлы после работы с ними может привести к утечкам памяти и другим проблемам.

Также существуют более удобные способы работы с файлами в Python, такие как использование оператора with. Оператор with автоматически закрывает файл после завершения блока кода, что упрощает управление файлами. Например:

```
with open('file.txt', 'r') as file:
    content = file.read()
```

В этом примере файл автоматически закрывается после завершения блока with, даже если в нем возникнет исключение.

Кроме того, в Python существуют различные функции и методы для работы с файлами, такие как write(), readlines(), seek() и т.д.

Метод write() используется для записи данных в файл. Он принимает один аргумент - строку, которую нужно записать. Если файл уже существует, то запись происходит в конец файла. Если файл не существует, то он будет создан. Вот пример использования:

```
with open('file.txt', 'w') as file:
      file.write('Hello, world!\n')
```

Метод readlines() используется для чтения всех строк из файла в виде списка строк. Вот пример использования:

```
with open('file.txt', 'r') as file:    lines = file.readlines()
      for line in lines:
      print(line)
```

Метод seek() используется для изменения текущей позиции в файле. Он принимает один аргумент - позицию, куда нужно переместить указатель файла. Позиция может быть задана как от начала файла, так и от текущей позиции. Вот пример использования:

```
with open('file.txt', 'r') as file:
      # перемещаем указатель в конец файла
      file.seek(0, 2)
      # записываем новую строку в файл
      file.write('Another line\n')
      # перемещаем указатель в начало файла
      file.seek(0)
      # читаем все строки из файла
      lines = file.readlines()
      for line in lines:
          print(line)
```

Кроме чтения и записи файлов, существуют также другие операции, которые можно выполнить с файлами в Python:

Переименование и удаление файлов.
Для переименования или удаления файлов в Python используется функция os.rename() для переименования и функция os.remove() для удаления файла. Например:

```
import os

# переименование файла
os.rename("old_file.txt", "new_file.txt")
```

```
# удаление файла
os.remove("file_to_delete.txt")
```

Проверка существования файла
Для проверки существования файла в Python можно использовать функцию os.path.exists(). Например:

```
import os

if os.path.exists("file_to_check.txt"):
      print("Файл существует")
else:
      print("Файл не существует")
```

Создание директорий
Для создания директории в Python можно использовать функцию os.mkdir(). Например:

```
import os

os.mkdir("new_directory")
```

Получение списка файлов и директорий
Для получения списка файлов и директорий в текущей директории можно использовать функцию os.listdir(). Например:

```
import os

files = os.listdir()
print(files)
```

Также можно указать путь к директории, для которой нужно получить список файлов и директорий.

```
import os

files = os.listdir("C:/Users/user/Documents")
print(files)
```

**Работа с JSON:**

В Python для работы с форматом данных JSON (JavaScript Object Notation) используется стандартный модуль json. Этот модуль позволяет преобразовывать объекты Python в формат JSON и наоборот, парсить JSON-данные и создавать из них объекты Python.

Для начала импортируем модуль json:

```
import json
```

**Кодирование объекта в JSON**:
Для преобразования объекта Python в формат JSON используется метод json.dumps():

```
import json

data = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

json_string = json.dumps(data)

print(json_string)
```

Результат выполнения:

```
{"name": "John", "age": 30, "city": "New York"}
```

**Декодирование JSON-данных в объект Python**:
Для преобразования JSON-данных в объект Python используется метод json.loads():

```
import json

json_string = '{"name": "John", "age": 30, "city": "New York"}'

data = json.loads(json_string)

print(data)
```

Результат выполнения:

```
{'name': 'John', 'age': 30, 'city': 'New York'}
```

**Работа с файлами JSON**:
Для чтения данных из файла JSON можно использовать метод json.load():

```
import json

with open('data.json', 'r') as f:
    data = json.load(f)
print(data)
```

Для записи данных в файл JSON можно использовать метод json.dump():

```
import json

data = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

with open('data.json', 'w') as f:
    json.dump(data, f)
```