Python - Programação CGI

A Common Gateway Interface, ou CGI, é um conjunto de padrões que define como as informações são trocadas entre o servidor web e um script personalizado. As especificações CGI são atualmente mantidas pela NCSA.

O que é CGI?

- A Common Gateway Interface, ou CGI, é um padrão para programas de gateway externos fazerem interface com servidores de informações, como servidores HTTP.
- A versão atual é CGI/1.1 e CGI/1.2 está em andamento.

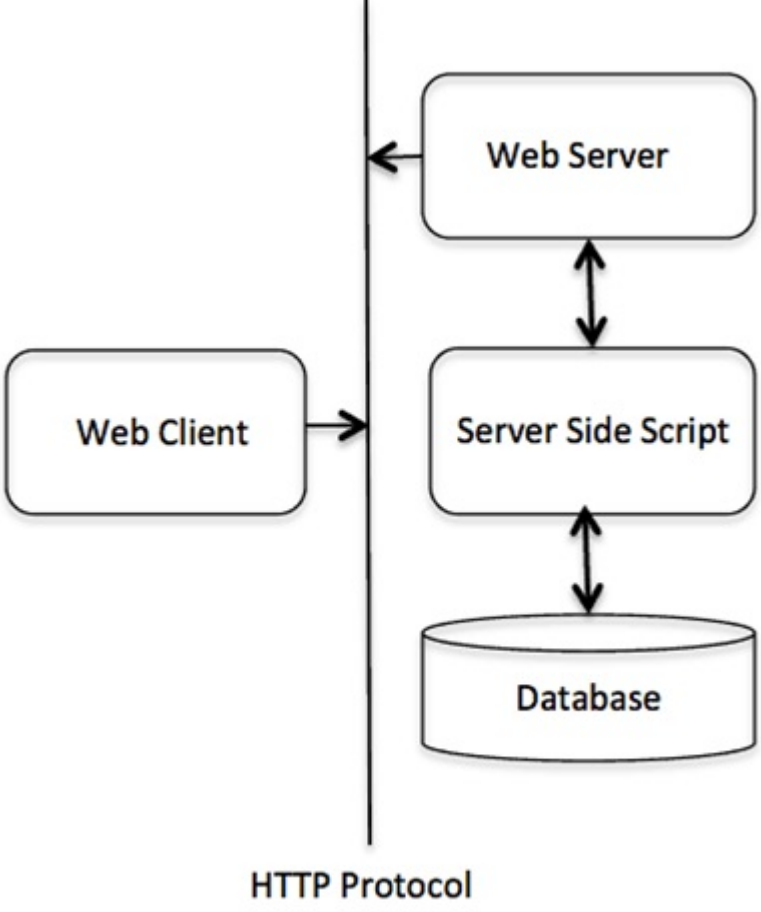
Navegação na Web

Para entender o conceito de CGI, vamos ver o que acontece quando clicamos em um hiperlink para navegar em uma determinada página da web ou URL.

- Seu navegador entra em contato com o servidor web HTTP e exige a URL, ou seja, o nome do arquivo.
- O Web Server analisa a URL e procura o nome do arquivo. Se encontrar esse arquivo, ele o envia de volta ao navegador; caso contrário, envia uma mensagem de erro indicando que você solicitou um arquivo errado.
- O navegador da Web recebe a resposta do servidor da Web e exibe o arquivo recebido ou a mensagem de erro.

Porém, é possível configurar o servidor HTTP para que sempre que um arquivo de um determinado diretório for solicitado, esse arquivo não seja enviado de volta; em vez disso, ele é executado como um programa e tudo o que o programa produz é enviado de volta para o seu navegador exibir. Esta função é chamada Common Gateway Interface ou CGI e os programas são chamados de scripts CGI. Esses programas CGI podem ser Python Script, PERL Script, Shell Script, programa C ou C++, etc.

Diagrama de Arquitetura CGI



Suporte e configuração de servidor web

Antes de prosseguir com a programação CGI, certifique-se de que seu servidor Web suporte CGI e esteja configurado para lidar com programas CGI. Todos os Programas CGI a serem executados pelo servidor HTTP são mantidos em um diretório pré-configurado. Este diretório é chamado CGI Directory e por convenção é nomeado como /var/www/cgi-bin. Por convenção, os arquivos CGI têm extensão como. **cgi,** mas você também pode manter seus arquivos com extensão python **.py .**

Por padrão, o servidor Linux está configurado para executar apenas os scripts no diretório cgi-bin em /var/www. Se você deseja especificar qualquer outro diretório para executar seus scripts CGI, comente as seguintes linhas no arquivo httpd.conf -

```
<Directory "/var/www/cgi-bin">
   AllowOverride None
   Options ExecCGI
   Order allow,deny
   Allow from all
</Directory>

<Directory "/var/www/cgi-bin">
Options All
</Directory>
```

A linha a seguir também deve ser adicionada para que o servidor Apache trate o arquivo .py como script cgi.

```
AddHandler cgi-script .py
```

Aqui, presumimos que você tenha o servidor Web instalado e funcionando com sucesso e seja capaz de executar qualquer outro programa CGI como Perl ou Shell, etc.

Primeiro programa CGI

Aqui está um link simples, vinculado a um script CGI chamado hello.py . Este arquivo é mantido no diretório /var/www/cgi-bin e possui o seguinte conteúdo. Antes de executar seu programa CGI, certifique-se de alterar o modo do arquivo usando o comando **chmod 755 hello.py** UNIX para tornar o arquivo executável.

```
print ("Content-type:text/html\r\n\r\n")
print ('<html>')
print ('<head>')
print ('<title>Hello Word - First CGI Program</title>')
print ('</head>')
print ('<body>')
print ('<h2>Hello Word! This is my first CGI program</h2>')
print ('</body>')
print ('</html>')
```

Note - A primeira linha do script deve ser o caminho para o executável Python. Aparece como um comentário no programa Python, mas é chamado de linha shebang.

No Linux, deve ser #!/usr/bin/python3.

No Windows, deve ser #!c:/python311/python.exd.

Digite o seguinte URL em seu navegador -

```
http://localhost/cgi-bin/hello.py
```

Hello Word! This is my first CGI program

Este script hello.py é um script Python simples, que grava sua saída no arquivo STDOUT, ou seja, na tela. Há um recurso importante e extra disponível que é a primeira linha a ser impressa **Content-type:text/html\r\n\r\n** . Esta linha é enviada de volta ao navegador e especifica o tipo de conteúdo a ser exibido na tela do navegador.

Até agora você deve ter entendido o conceito básico de CGI e pode escrever muitos programas CGI complicados usando Python. Este script pode interagir com qualquer outro sistema externo também para trocar informações como RDBMS.

Cabeçalho HTTP

A linha **Content-type:text/html\r\n\r\n** faz parte do cabeçalho HTTP que é enviado ao navegador para entender o conteúdo. Todo o cabeçalho HTTP estará no seguinte formato -

```
HTTP Field Name: Field Content

For Example
Content-type: text/html\r\n\r\n
```

Existem alguns outros cabeçalhos HTTP importantes, que você usará com frequência em sua programação CGI.

Sr. Não.	Cabeçalho e descrição
1	**Tipo de conteúdo:** Uma string MIME que define o formato do arquivo que está sendo retornado. O exemplo é Content-type:text/html
2	**Expira em: Data** A data em que a informação se torna inválida. É usado pelo navegador para decidir quando uma página precisa ser atualizada. Uma sequência de data válida está no formato 01 de janeiro de 1998 12:00:00 GMT.
3	**Localização: URL** O URL retornado em vez do URL solicitado. Você pode usar este campo para redirecionar uma solicitação para qualquer arquivo.
4	**Última modificação: Data** A data da última modificação do recurso.
5	**Comprimento do conteúdo: N** O comprimento, em bytes, dos dados que estão sendo retornados. O navegador usa esse valor para relatar o tempo estimado de download de um arquivo.
6	**Set-Cookie: String** Defina o cookie passado pela string

Variáveis de ambiente CGI

Todos os programas CGI têm acesso às seguintes variáveis de ambiente. Essas variáveis desempenham um papel importante ao escrever qualquer programa CGI.

Sr. Não.	Nome e descrição da variável
1	**TIPO DE CONTEÚDO** O tipo de dados do conteúdo. Usado quando o cliente está enviando conteúdo anexado ao servidor. Por exemplo, upload de arquivo.
2	**COMPRIMENTO DO CONTEÚDO** O comprimento das informações da consulta. Está disponível apenas para solicitações POST.
3	**HTTP_COOKIE** Retorna os cookies definidos na forma de par chave e valor.
4	**HTTP_USER_AGENT** O campo do cabeçalho da solicitação User-Agent contém informações sobre o agente do usuário que originou a solicitação. É o nome do navegador da web.
5	**PATH_INFO** O caminho para o script CGI.
6	**QUERY_STRING** As informações codificadas em URL que são enviadas com a solicitação do método GET.
7	**REMOTE_ADDR** O endereço IP do host remoto que faz a solicitação. Este é um registro útil ou para autenticação.
8	**HOSPEDEIRO REMOTO** O nome totalmente qualificado do host que faz a solicitação. Se esta informação não estiver disponível, então REMOTE_ADDR poderá ser usado para obter o endereço IR.
9	**REQUEST_METHOD** O método usado para fazer a solicitação. Os métodos mais comuns são GET e POST.

10	**SCRIPT_FILENAME** O caminho completo para o script CGI.
11	**SCRIPT_NAME** O nome do script CGI.
12	**NOME DO SERVIDOR** O nome do host ou endereço IP do servidor
13	**SERVIDOR_SOFTWARE** O nome e a versão do software que o servidor está executando.

Aqui está um pequeno programa CGI para listar todas as variáveis CGI. Clique neste link para ver o resultado Obter Ambiente

```
import os

print ("Content-type: text/html\r\n\r\n");
print ("<font size=+1>Environment</font><\br>");
for param in os.environ.keys():
   print ("<b>%20s</b>: %s<\br>" % (param, os.environ[param]))
```

Métodos GET e POST

Você deve ter se deparado com muitas situações em que precisa passar algumas informações do seu navegador para o servidor web e, finalmente, para o seu programa CGI. Mais frequentemente, o navegador usa dois métodos para passar essas informações ao servidor web. Esses métodos são Método GET e Método POST.

Passando informações usando o método GET

O método GET envia as informações codificadas do usuário anexadas à solicitação da página. A página e as informações codificadas são separadas pelo ? personagem da seguinte forma -

```
http://www.test.com/cgi-bin/hello.py?key1=value1&key2=value2
```

- O método GET é o método padrão para passar informações do navegador para o servidor web e produz uma longa string que aparece na caixa Location: do seu navegador.
- Nunca use o método GET se você tiver senha ou outras informações confidenciais para passar ao servidor.
- O método GET possui limite de tamanho: apenas 1024 caracteres podem ser enviados em uma string de solicitação.

- O método GET envia informações utilizando o cabeçalho QUERY_STRING e estará acessível em seu Programa CGI através da variável de ambiente QUERY_STRING.

Você pode passar informações simplesmente concatenando pares de chave e valor junto com qualquer URL ou pode usar tags HTML <FORM> para passar informações usando o método GET.

Exemplo de URL simples: método Get

Aqui está uma URL simples, que passa dois valores para o programa hello_get.py usando o método GET.

```
/cgi-bin/hello_get.py?first_name=Malhar&last_name=Lathkar
```

Abaixo está o script **hello_get.py** para lidar com a entrada fornecida pelo navegador da web. Vamos utilizar o módulo cgi, que facilita muito o acesso às informações passadas -

```
# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
first_name = form.getvalue('first_name')
last_name = form.getvalue('last_name')

print ("Content-type:text/html")
print()
print ("<html>")
print ('<head>')
print ("<title>Hello - Second CGI Program</title>")
print ('</head>')
print ('<body>')
print ("<h2>Hello %s %s</h2>" % (first_name, last_name))
print ('</body>')
print ('</html>')
```

Isso geraria o seguinte resultado -

Hello Malhar Lathkar

Exemplo de FORM simples: Método GET

Este exemplo passa dois valores usando HTML FORM e botão enviar. Usamos o mesmo script CGI hello_get.py para lidar com essa entrada.

```
<form action = "/cgi-bin/hello_get.py" method = "get">
   First Name: <input type = "text" name = "first_name">  <br />

   Last Name: <input type = "text" name = "last_name" />
   <input type = "submit" value = "Submit" />
</form>
```

Aqui está o resultado real do formulário acima: você insere o nome e o sobrenome e clica no botão enviar para ver o resultado.

Primeiro nome:
Sobrenome: Enviar

Passando informações usando o método POST

Um método geralmente mais confiável de passar informações para um programa CGI é o método POST. Isso empacota as informações exatamente da mesma maneira que os métodos GET, mas em vez de enviá-las como uma sequência de texto após um ? na URL ele envia como uma mensagem separada. Esta mensagem chega ao script CGI na forma de entrada padrão.

Abaixo está o mesmo script hello_get.py que lida com os métodos GET e POST.

```
# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
first_name = form.getvalue('first_name')
last_name  = form.getvalue('last_name')

print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>"
print "<title>Hello - Second CGI Program</title>"
print "</head>"
print "<body>"
print "<h2>Hello %s %s</h2>" % (first_name, last_name)
```

```
print "</body>"
print "</html>"
```

Tomemos novamente o mesmo exemplo acima, que passa dois valores usando HTML FORM e botão de envio. Usamos o mesmo script CGI hello_get.py para lidar com essa entrada.

```
<form action = "/cgi-bin/hello_get.py" method = "post">
First Name: <input type = "text" name = "first_name"><br />
Last Name: <input type = "text" name = "last_name" />

<input type = "submit" value = "Submit" />
</form>
```

Aqui está a saída real do formulário acima. Você insere o nome e o sobrenome e clica no botão enviar para ver o resultado.

Primeiro nome:
Sobrenome: Enviar

Passando dados da caixa de seleção para o programa CGI

As caixas de seleção são usadas quando é necessário selecionar mais de uma opção.

Aqui está um exemplo de código HTML para um formulário com duas caixas de seleção -

```
<form action = "/cgi-bin/checkbox.cgi" method = "POST" target = "_blank">
   <input type = "checkbox" name = "maths" value = "on" /> Maths
   <input type = "checkbox" name = "physics" value = "on" /> Physics
   <input type = "submit" value = "Select Subject" />
</form>
```

O resultado deste código é o seguinte formato -

☐Matemáticas ☐Física Selecione Assunto

Abaixo está o script checkbox.cgi para lidar com a entrada fornecida pelo navegador da web para o botão da caixa de seleção.

```
# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
if form.getvalue('maths'):
   math_flag = "ON"
```

```
else:
   math_flag = "OFF"

if form.getvalue('physics'):
   physics_flag = "ON"
else:
   physics_flag = "OFF"

print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>"
print "<title>Checkbox - Third CGI Program</title>"
print "</head>"
print "<body>"
print "<h2> CheckBox Maths is : %s</h2>" % math_flag
print "<h2> CheckBox Physics is : %s</h2>" % physics_flag
print "</body>"
print "</html>"
```

Passando dados do botão de opção para o programa CGI

Os botões de opção são usados quando apenas uma opção precisa ser selecionada.

Aqui está um exemplo de código HTML para um formulário com dois botões de opção -

```
<form action = "/cgi-bin/radiobutton.py" method = "post" target = "_blank">
   <input type = "radio" name = "subject" value = "maths" /> Maths
   <input type = "radio" name = "subject" value = "physics" /> Physics
   <input type = "submit" value = "Select Subject" />
</form>
```

O resultado deste código é o seguinte formato -

○Matemáticas ○Física Selecione Assunto

Abaixo está o script radiobutton.py para lidar com a entrada fornecida pelo navegador da web para o botão de opção -

```
# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
if form.getvalue('subject'):
   subject = form.getvalue('subject')
```

```
else:
   subject = "Not set"

print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>"
print "<title>Radio - Fourth CGI Program</title>"
print "</head>"
print "<body>"
print "<h2> Selected Subject is %s</h2>" % subject
print "</body>"
print "</html>"
```

Passando dados da área de texto para o programa CGI

O elemento TEXTAREA é usado quando texto multilinha deve ser passado para o programa CGI.

Aqui está um exemplo de código HTML para um formulário com uma caixa TEXTAREA -

```
<form action = "/cgi-bin/textarea.py" method = "post" target = "_blank">
   <textarea name = "textcontent" cols = "40" rows = "4">
      Type your text here...
   </textarea>
   <input type = "submit" value = "Submit" />
</form>
```

O resultado deste código é o seguinte formato -

Type your text here...

Enviar

Abaixo está o script textarea.cgi para lidar com as entradas fornecidas pelo navegador da web -

```
# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
if form.getvalue('textcontent'):
   text_content = form.getvalue('textcontent')
```

```
else:
   text_content = "Not entered"

print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>";
print "<title>Text Area - Fifth CGI Program</title>"
print "</head>"
print "<body>"
print "<h2> Entered Text Content is %s</h2>" % text_content
print "</body>"
```

Passando dados da caixa suspensa para o programa CGI

A caixa suspensa é usada quando temos muitas opções disponíveis, mas apenas uma ou duas serão selecionadas.

Aqui está um exemplo de código HTML para um formulário com uma caixa suspensa -

```
<form action = "/cgi-bin/dropdown.py" method = "post" target = "_blank">
   <select name = "dropdown">
      <option value = "Maths" selected>Maths</option>
      <option value = "Physics">Physics</option>
   </select>
   <input type = "submit" value = "Submit"/>
</form>
```

O resultado deste código é o seguinte formato -

Matemáticas Enviar

Abaixo está o script dropdown.py para lidar com as entradas fornecidas pelo navegador da web.

```
# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
if form.getvalue('dropdown'):
   subject = form.getvalue('dropdown')
else:
   subject = "Not entered"

print "Content-type:text/html\r\n\r\n"
```

```
print "<html>"
print "<head>"
print "<title>Dropdown Box - Sixth CGI Program</title>"
print "</head>"
print "<body>"
print "<h2> Selected Subject is %s</h2>" % subject
print "</body>"
print "</html>"
```

Usando Cookies em CGI

O protocolo HTTP é um protocolo sem estado. Para um site comercial, é necessário manter as informações da sessão entre as diferentes páginas. Por exemplo, o registro de um usuário termina após a conclusão de muitas páginas. Como manter as informações da sessão do usuário em todas as páginas da web?

Em muitas situações, o uso de cookies é o método mais eficiente de lembrar e rastrear preferências, compras, comissões e outras informações necessárias para uma melhor experiência do visitante ou estatísticas do site.

Como funciona?

Seu servidor envia alguns dados ao navegador do visitante na forma de um cookie. O navegador pode aceitar o cookie. Se isso acontecer, ele será armazenado como um registro de texto simples no disco rígido do visitante. Agora, quando o visitante chega em outra página do seu site, o cookie fica disponível para recuperação. Uma vez recuperado, seu servidor sabe/lembra o que foi armazenado.

Os cookies são um registro de dados de texto simples de 5 campos de comprimento variável -

- **Expires** - A data em que o cookie irá expirar. Se estiver em branco, o cookie expirará quando o visitante sair do navegador.
- **Domain** - O nome de domínio do seu site.
- **Path** - O caminho para o diretório ou página da web que define o cookie. Pode ficar em branco se você quiser recuperar o cookie de qualquer diretório ou página.
- **Secure** - Se este campo contiver a palavra “seguro”, então o cookie só poderá ser recuperado com um servidor seguro. Se este campo estiver em branco, tal restrição não existe.
- **Name = Value** - Os cookies são definidos e recuperados na forma de pares de chave e valor.

Configurando Cookies

É muito fácil enviar cookies para o navegador. Esses cookies são enviados junto com o cabeçalho HTTP antes do campo Content-type. Supondo que você queira definir UserID e Password como cookies. A configuração dos cookies é feita da seguinte forma -

```
print "Set-Cookie:UserID = XYZ;\r\n"
print "Set-Cookie:Password = XYZ123;\r\n"
print "Set-Cookie:Expires = Tuesday, 31-Dec-2007 23:12:40 GMT;\r\n"
print "Set-Cookie:Domain = www.tutorialspoint.com;\r\n"
print "Set-Cookie:Path = /perl;\n"
print "Content-type:text/html\r\n\r\n"
.....Rest of the HTML Content....
```

A partir deste exemplo, você deve ter entendido como definir cookies. Usamos o cabeçalho HTTP **Set-Cookie** para definir cookies.

É opcional definir atributos de cookies como Expira, Domínio e Caminho. É notável que os cookies são definidos antes do envio da linha mágica **"Content-type:text/html\r\n\r\n** .

Recuperando Cookies

É muito fácil recuperar todos os cookies definidos. Os cookies são armazenados na variável de ambiente CGI HTTP_COOKIE e terão o seguinte formato -

```
key1 = value1;key2 = value2;key3 = value3....
```

Aqui está um exemplo de como recuperar cookies.

```
# Import modules for CGI handling
from os import environ
import cgi, cgitb

if environ.has_key('HTTP_COOKIE'):
   for cookie in map(strip, split(environ['HTTP_COOKIE'], ';')):
      (key, value ) = split(cookie, '=');
      if key == "UserID":
         user_id = value

      if key == "Password":
         password = value

print "User ID  = %s" % user_id
print "Password = %s" % password
```

Isso produz o seguinte resultado para os cookies definidos pelo script acima -

User ID = XYZ
Password = XYZ123

Exemplo de upload de arquivo

Para fazer upload de um arquivo, o formulário HTML deve ter o atributo enctype definido como **multipart/form-data** . A tag de entrada com o tipo de arquivo cria um botão "Navegar".

```
<html>
   <body>
      <form enctype = "multipart/form-data" action = "save_file.py" method = "post">
      <p>File: <input type = "file" name = "filename" /></p>
      <p><input type = "submit" value = "Upload" /></p>
      </form>
   </body>
</html>
```

O resultado deste código é o seguinte formato -

Arquivo: Escolher arquivo Nenhum arquivo escolhido
Upload

O exemplo acima foi desabilitado intencionalmente para evitar que as pessoas carreguem arquivos em nosso servidor, mas você pode tentar o código acima com seu servidor.

Aqui está o script **save_file.py** para lidar com o upload do arquivo -

```
import cgi, os
import cgitb; cgitb.enable()

form = cgi.FieldStorage()

# Get filename here.
fileitem = form['filename']

# Test if the file was uploaded
if fileitem.filename:
   # strip leading path from file name to avoid
   # directory traversal attacks
   fn = os.path.basename(fileitem.filename)
   open('/tmp/' + fn, 'wb').write(fileitem.file.read())

   message = 'The file "' + fn + '" was uploaded successfully'
```

```
else:
   message = 'No file was uploaded'

print """\
Content-Type: text/html\n
<html>
   <body>
      <p>%s</p>
   </body>
</html>
""" % (message,)
```

Se você executar o script acima no Unix/Linux, precisará tomar o cuidado de substituir o separador de arquivos da seguinte maneira, caso contrário, em sua máquina Windows, a instrução open() acima deve funcionar bem.

```
fn = os.path.basename(fileitem.filename.replace("\\", "/" ))
```

Como abrir uma caixa de diálogo “Download de arquivo”?

Às vezes, é desejável que você dê a opção de um usuário clicar em um link e uma caixa de diálogo "Download de arquivo" será exibida para o usuário em vez de exibir o conteúdo real. Isso é muito fácil e pode ser conseguido através do cabeçalho HTTP. Este cabeçalho HTTP é diferente do cabeçalho mencionado na seção anterior.

Por exemplo, se você deseja disponibilizar um arquivo **FileName** para download a partir de um determinado link, sua sintaxe é a seguinte -

```
# HTTP Header
print "Content-Type:application/octet-stream; name = \"FileName\"\r\n";
print "Content-Disposition: attachment; filename = \"FileName\"\r\n\n";

# Actual File Content will go here.
fo = open("foo.txt", "rb")

str = fo.read();
print str

# Close opend file
fo.close()
```