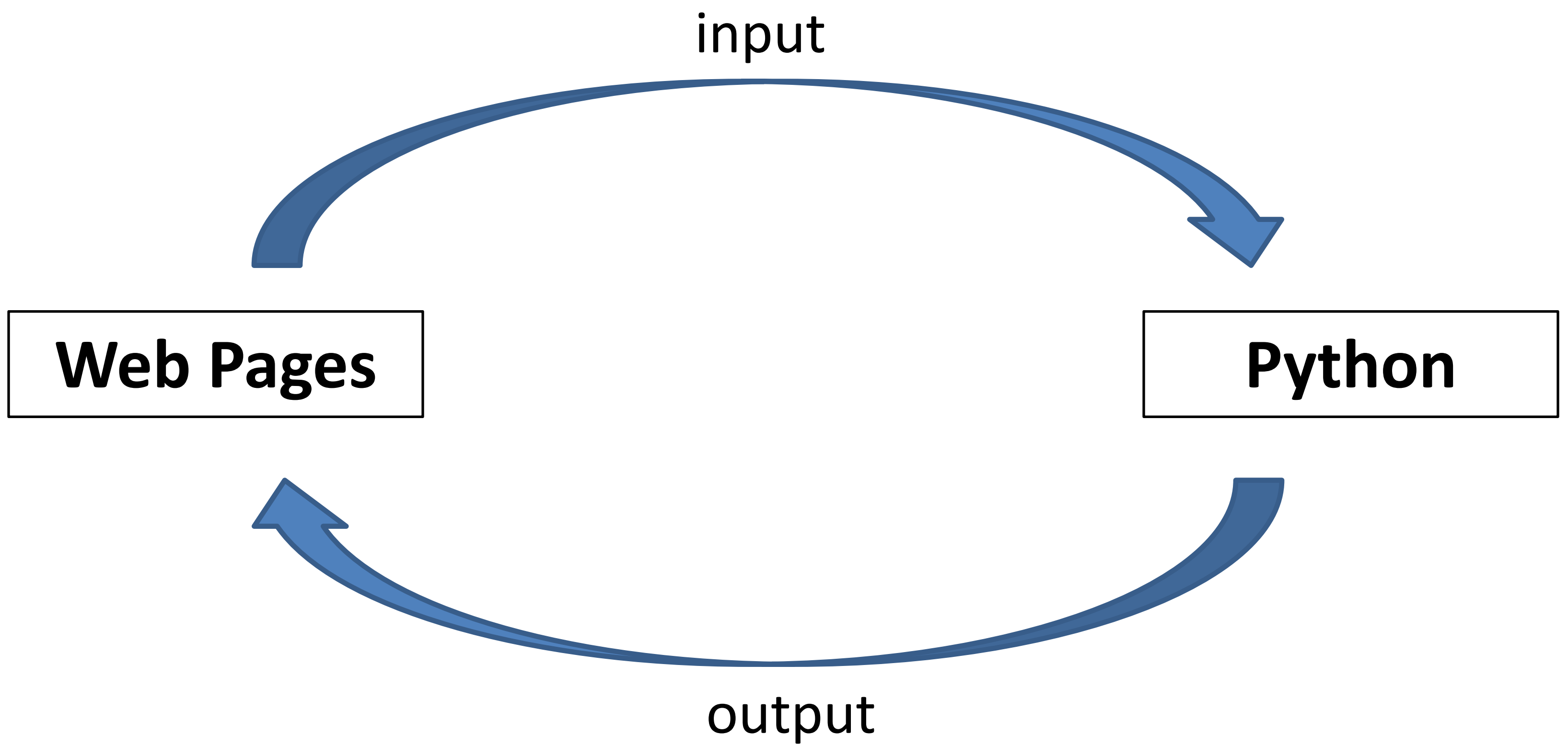
STSCI 4060

Lecture File 9

Xiaolong Yang
(xy44@cornell.edu)

Creating Dynamic Web Pages with Python

Python and Web Pages



- Webpages provide data to a Python program (a Python server program).
- The Python program transforms the input data into desired output.
- The output is embedded in a dynamic web page, which is displayed to the users.

A Few Bits About Web Pages

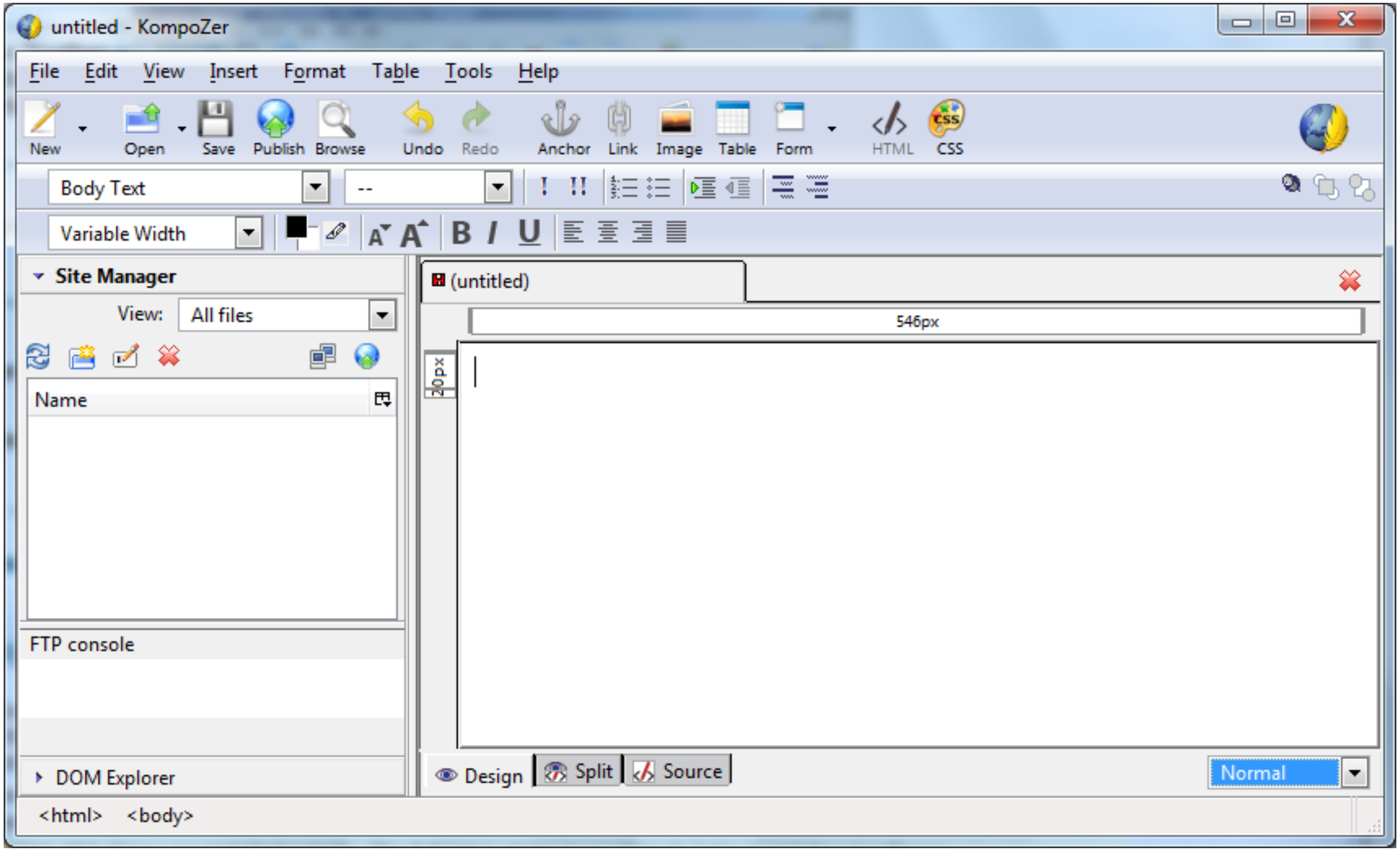
- Web pages are formatted documents, that are marked up with various tags using the hypertext markup language (HTML). All HTML markup is delimited by tags enclosed in angle brackets, most of which come in pairs, surrounding the text to be formatted. Note that the end tag has an extra '/'. For example,

```
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type">
 <title>Hello</title>
</head>
<body>
Hello, World!
</body>
</html>
```

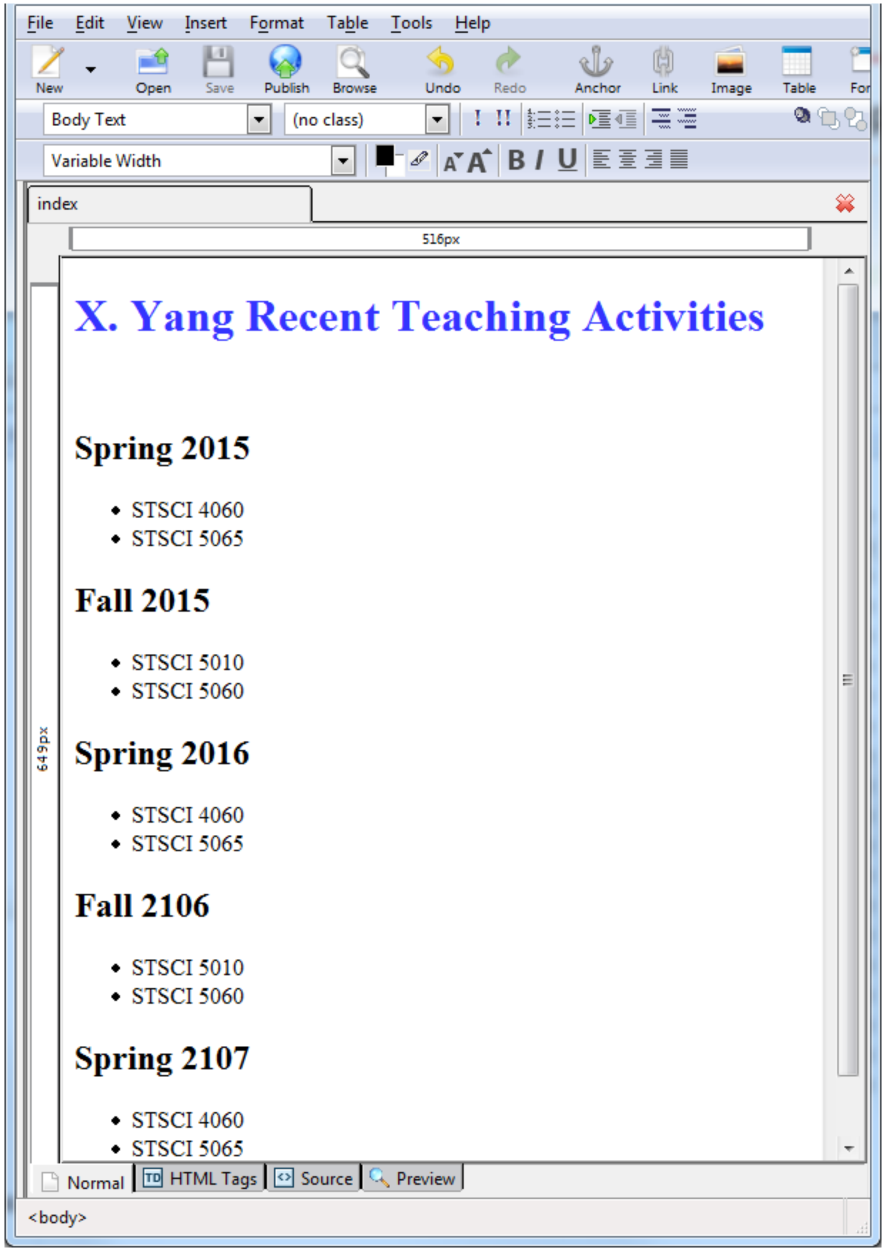
- You may use a plain text editor to produce a web page, but often people use some specialized editors, e.g., Dreamweaver, AceHTML, Visual Site Designer, Expression Studio, Kompozer, …
- We will use the open source KompoZer (downloadable from Blackboard under the Software folder).

Kompozer Basics

- Open your Kompozer software and create a new document by going to the File menu and clicking on New. You see the following window.



Kompozer Simple Demo …



X. Yang Recent Teaching Activities

Spring 2015

- STSCI 4060
- STSCI 5065

Fall 2015

- STSCI 5010
- STSCI 5060

Spring 2016

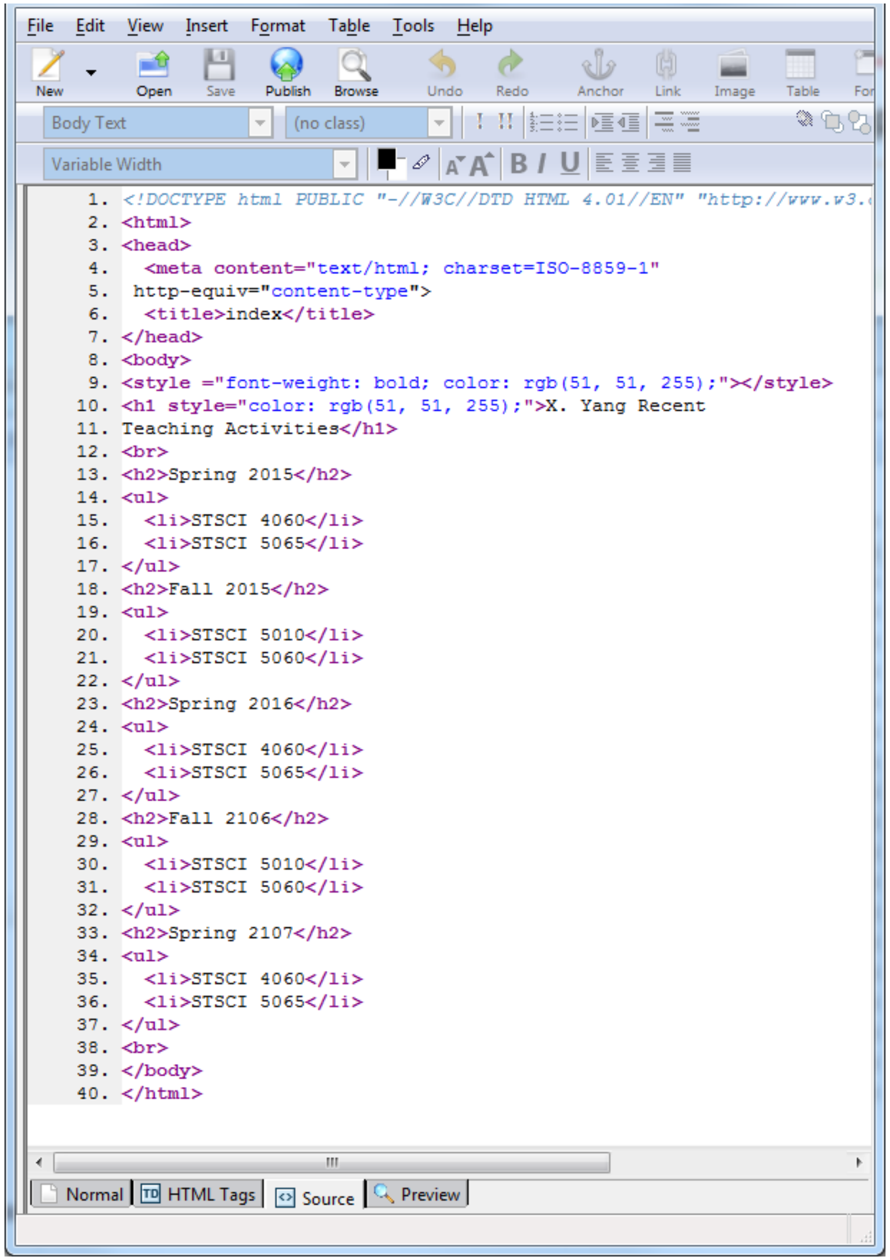
- STSCI 4060
- STSCI 5065

Fall 2106

- STSCI 5010
- STSCI 5060

Spring 2107

- STSCI 4060
- STSCI 5065



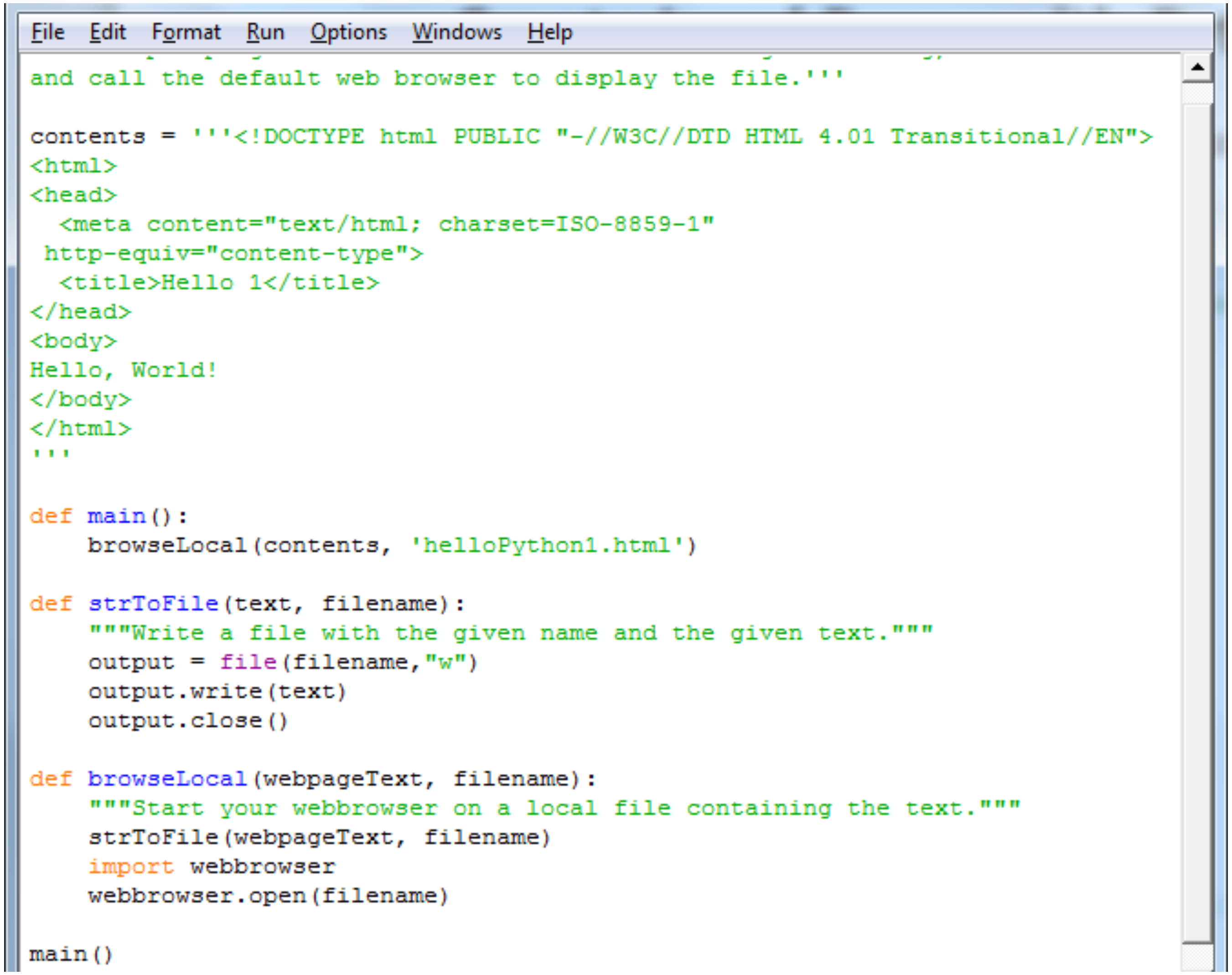
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type">
  <title>index</title>
</head>
<body>
<style ="font-weight: bold; color: rgb(51, 51, 255);"></style>
<h1 style="color: rgb(51, 51, 255);">X. Yang Recent
Teaching Activities</h1>
<br>
<h2>Spring 2015</h2>
<ul>
  <li>STSCI 4060</li>
  <li>STSCI 5065</li>
</ul>
<h2>Fall 2015</h2>
<ul>
  <li>STSCI 5010</li>
  <li>STSCI 5060</li>
</ul>
<h2>Spring 2016</h2>
<ul>
  <li>STSCI 4060</li>
  <li>STSCI 5065</li>
</ul>
<h2>Fall 2106</h2>
<ul>
  <li>STSCI 5010</li>
  <li>STSCI 5060</li>
</ul>
<h2>Spring 2107</h2>
<ul>
  <li>STSCI 4060</li>
  <li>STSCI 5065</li>
</ul>
<br>
</body>
</html>
```

Different Types of Documents Involved

- **Python files**
  - Files with .py extension: Regular Python programs to take input and produce output for displaying on a web page.
  - Files with .cgi extension: Python CGI (Common Gateway Interface) programs to be run from a web server. To run a CGI program for our situation, you must start at http://localhost:8081/, followed with the file name. This file must be in the same folder as the local server program, *localCGIServer.py*.
- **HTML files**
  - Template files and Output files: They are used by Python programs internally to create a template or format string to dynamically generate the final web pages.
  - Files for web browser display: These files all have a .html extension and should be stored in the same folder as the localCGIServer.py (which should be running all the time—run from the folder directly). In the browser URL field, the web page file must be preceded by http://localhost:8081/, e.g., http://localhost:8081/mypage.html.

Create Local Pages with Python

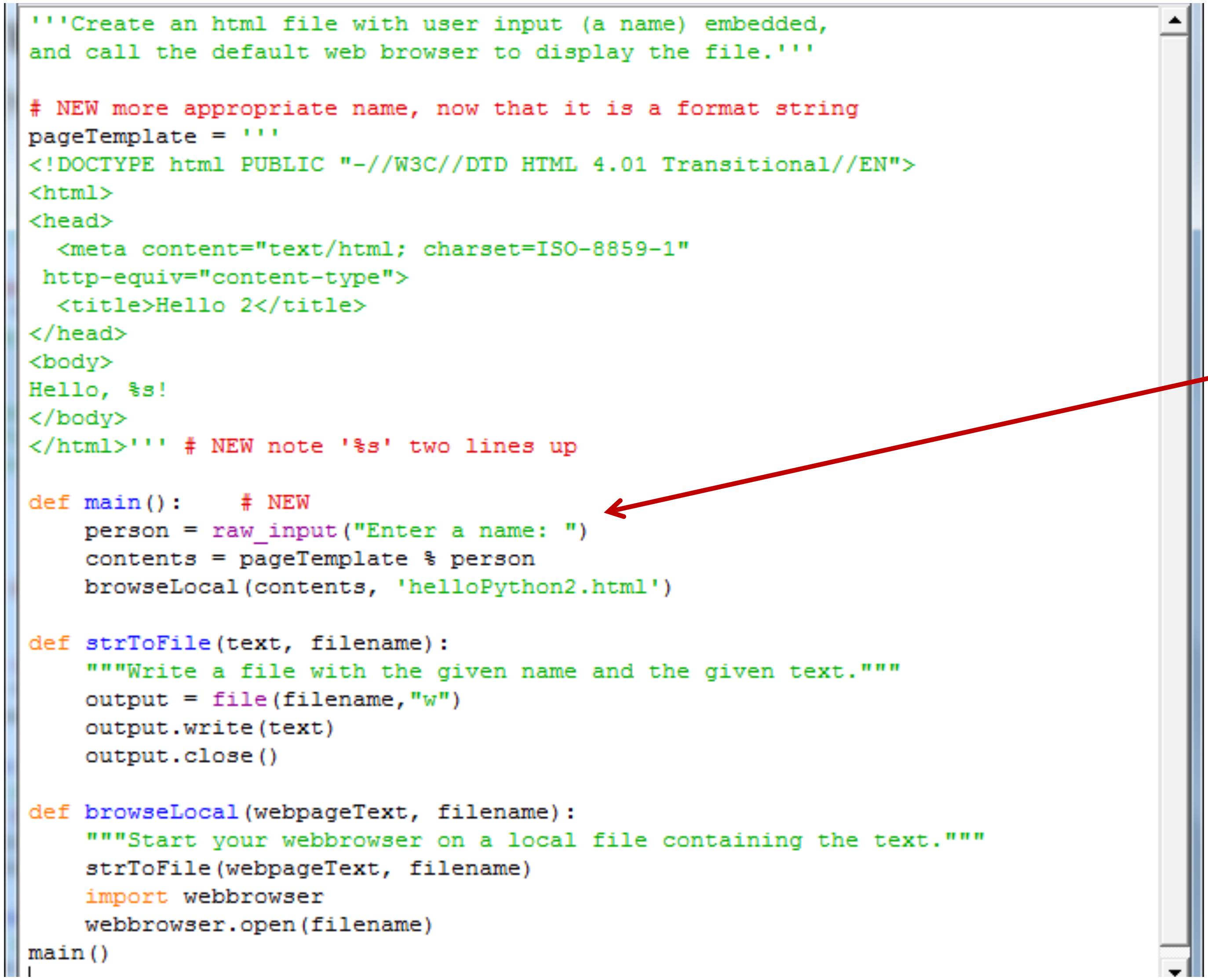
Version 1 (helloWeb1.py): The web page content is hard-coded in the program.



```
and call the default web browser to display the file.'''

contents = '''<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type">
  <title>Hello 1</title>
</head>
<body>
Hello, World!
</body>
</html>
'''

def main():
    browseLocal(contents, 'helloPython1.html')

def strToFile(text, filename):
    """Write a file with the given name and the given text."""
    output = file(filename,"w")
    output.write(text)
    output.close()

def browseLocal(webpageText, filename):
    """Start your webbrowser on a local file containing the text."""
    strToFile(webpageText, filename)
    import webbrowser
    webbrowser.open(filename)

main()
```

Create Local Pages with Python

Version 2 (helloWeb2.py): The web page content is hard-coded in the program but a formatted string is used.



```
'''Create an html file with user input (a name) embedded,
and call the default web browser to display the file.'''

# NEW more appropriate name, now that it is a format string
pageTemplate = '''
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type">
  <title>Hello 2</title>
</head>
<body>
Hello, %s!
</body>
</html>''' # NEW note '%s' two lines up

def main():    # NEW
    person = raw_input("Enter a name: ")
    contents = pageTemplate % person
    browseLocal(contents, 'helloPython2.html')

def strToFile(text, filename):
    """Write a file with the given name and the given text."""
    output = file(filename,"w")
    output.write(text)
    output.close()

def browseLocal(webpageText, filename):
    """Start your webbrowser on a local file containing the text."""
    strToFile(webpageText, filename)
    import webbrowser
    webbrowser.open(filename)
main()
```

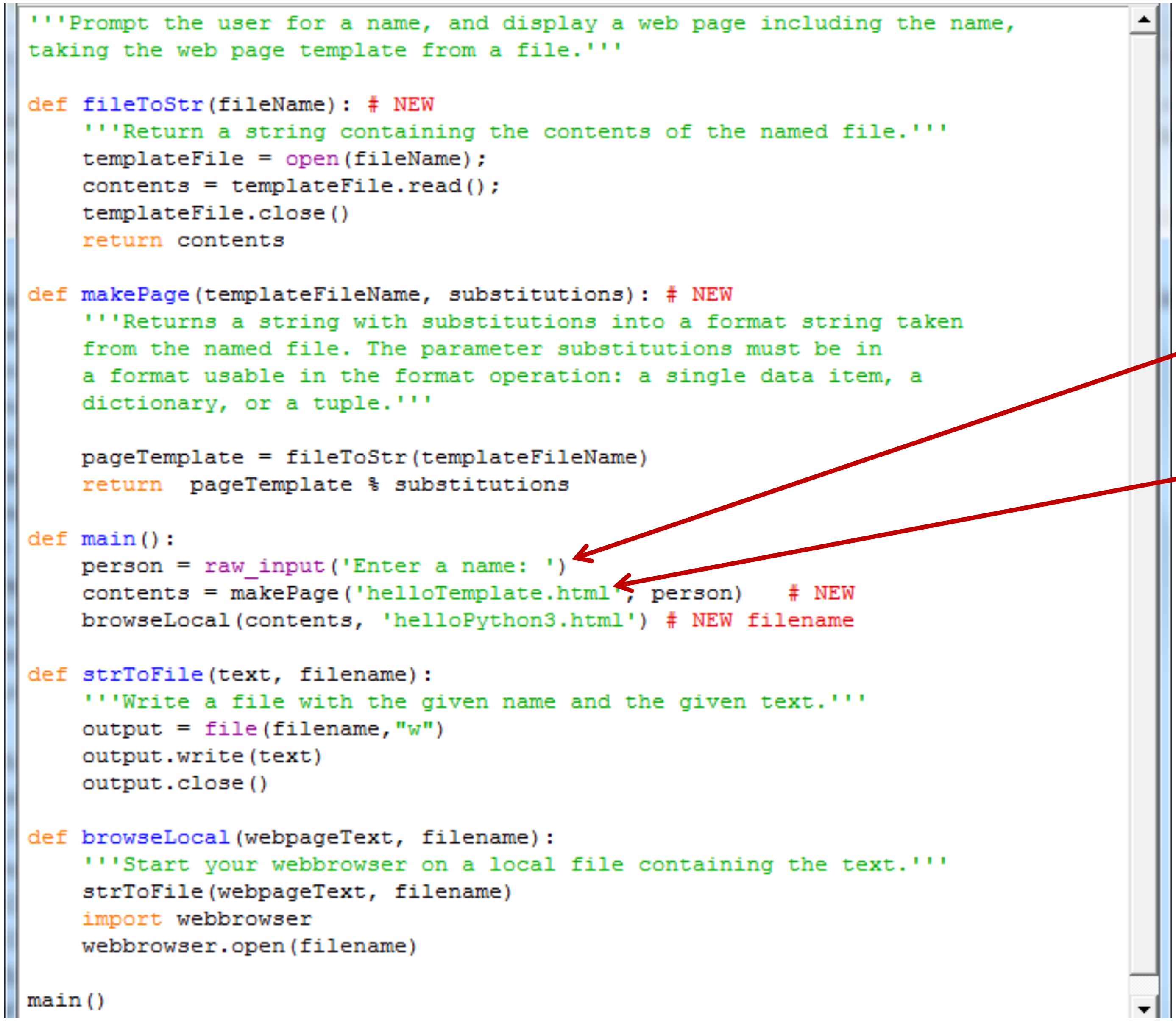
Interactive Input

Procedure: Dynamic Webpages Based on HTML Templates

$$\begin{tikzcd}
\boxed{\text{An HTML template, a file containing a format string(s)}} \arrow[d] \\
\boxed{\text{A string to be manipulated with Python}} \arrow[d] \\
\boxed{\text{A new HTML file}} \arrow[d] \\
\boxed{\text{Display on a web browser}}
\end{tikzcd}$$

Create Local Pages with Python

Version 3 (helloWeb3.py): The web page content is based on a **template** and **user input**.



```
'''Prompt the user for a name, and display a web page including the name,
taking the web page template from a file.'''

def fileToStr(fileName): # NEW
    '''Return a string containing the contents of the named file.'''
    templateFile = open(fileName);
    contents = templateFile.read();
    templateFile.close()
    return contents

def makePage(templateFileName, substitutions): # NEW
    '''Returns a string with substitutions into a format string taken
    from the named file. The parameter substitutions must be in
    a format usable in the format operation: a single data item, a
    dictionary, or a tuple.'''

    pageTemplate = fileToStr(templateFileName)
    return  pageTemplate % substitutions

def main():
    person = raw_input('Enter a name: ')
    contents = makePage('helloTemplate.html', person)    # NEW
    browseLocal(contents, 'helloPython3.html') # NEW filename

def strToFile(text, filename):
    '''Write a file with the given name and the given text.'''
    output = file(filename,"w")
    output.write(text)
    output.close()

def browseLocal(webpageText, filename):
    '''Start your webbrowser on a local file containing the text.'''
    strToFile(webpageText, filename)
    import webbrowser
    webbrowser.open(filename)

main()
```

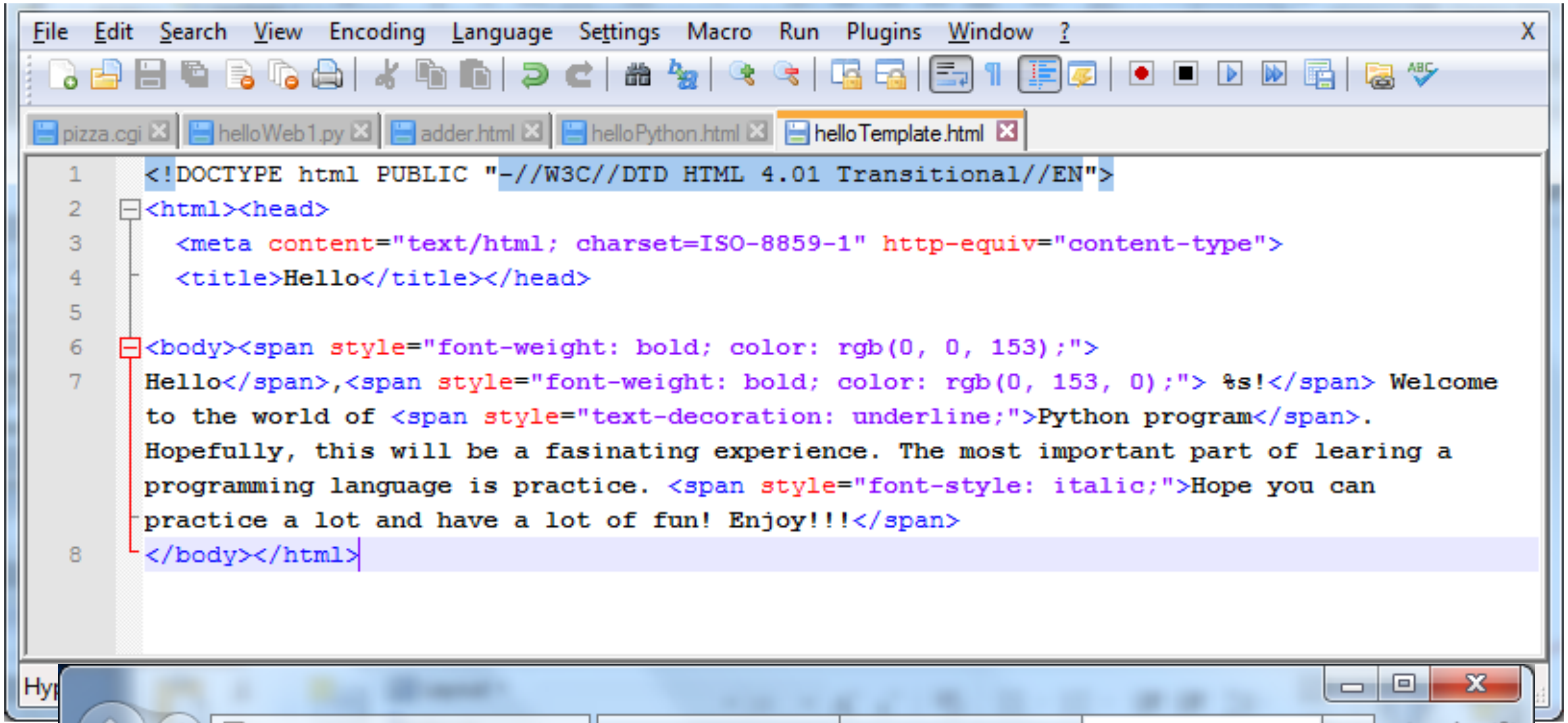
Interactive input

Template

Create Local Pages with Python

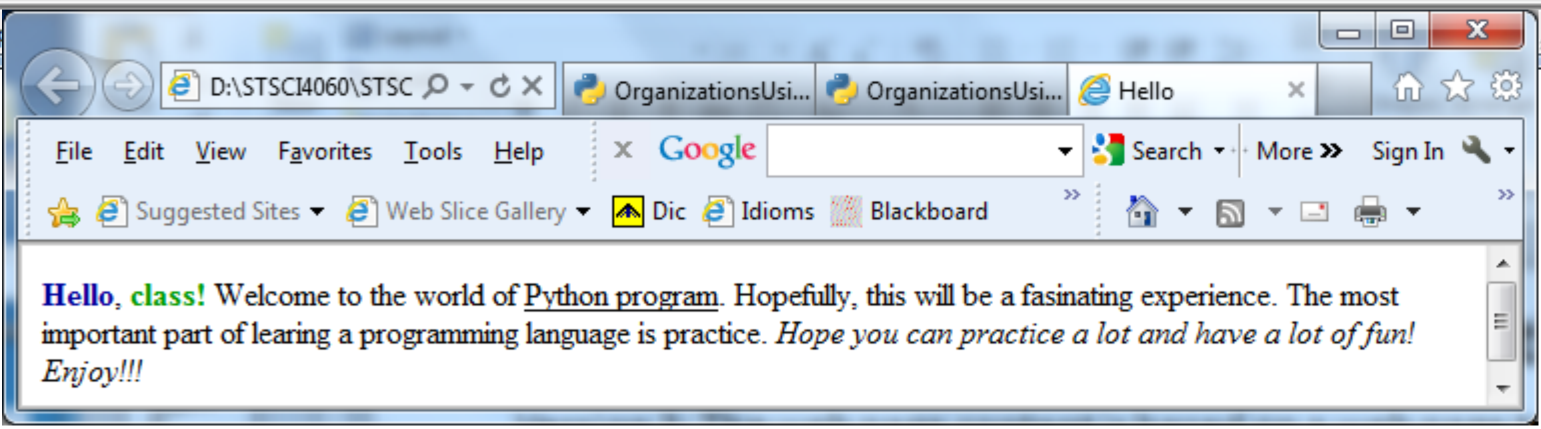
Version 3 (helloWeb3.py): The web page content is based on a template and user input.

Compose/modify the web page template in Notepad++ (or other editors)



File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
pizza.cgi helloWeb1.py adder.html helloPython.html helloTemplate.html

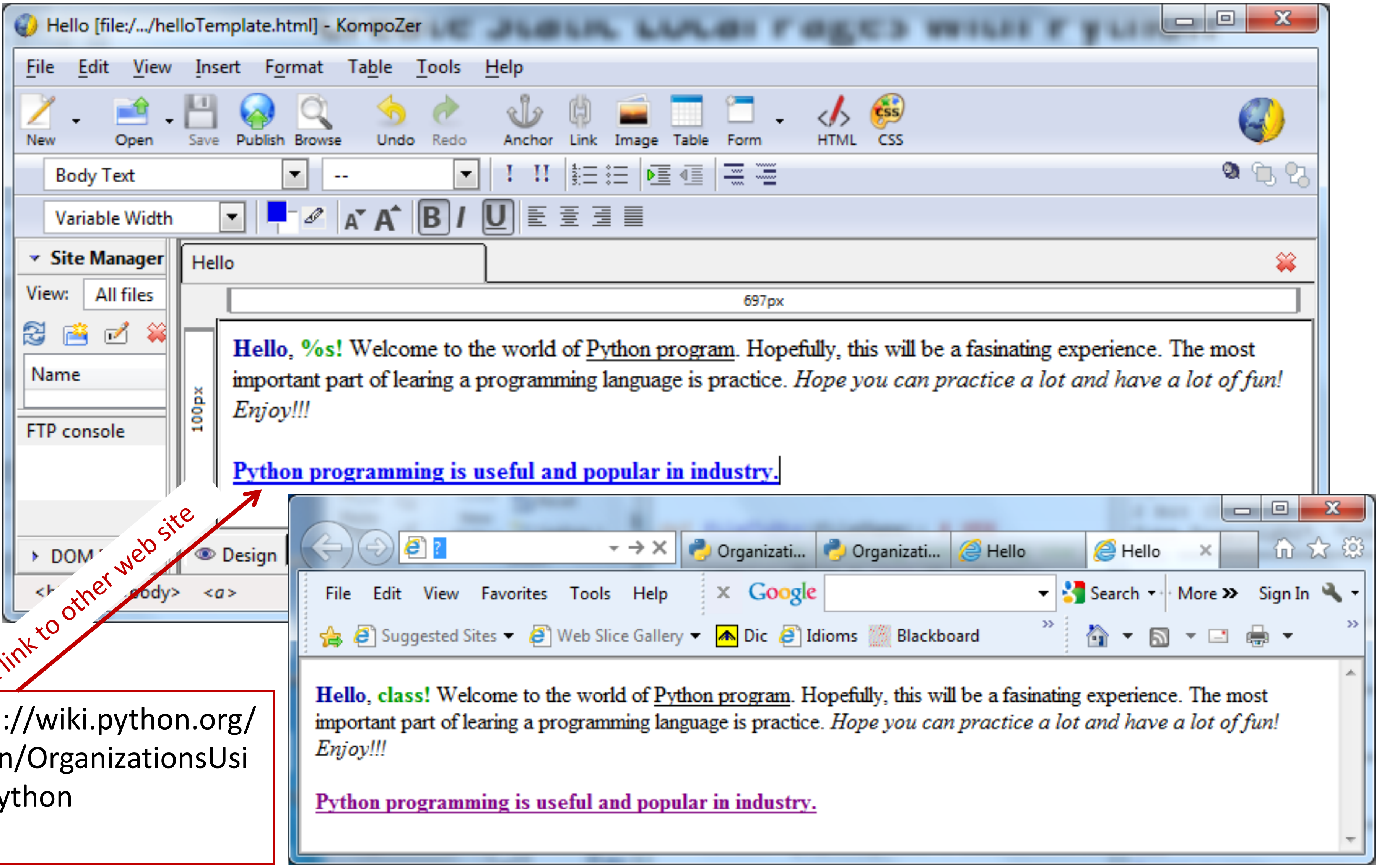
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
  <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
  <title>Hello</title></head>

<body><span style="font-weight: bold; color: rgb(0, 0, 153);">
Hello</span>,<span style="font-weight: bold; color: rgb(0, 153, 0);"> %s!</span> Welcome
to the world of <span style="text-decoration: underline;">Python program</span>.
Hopefully, this will be a fasinating experience. The most important part of learing a
programming language is practice. <span style="font-style: italic;">Hope you can
practice a lot and have a lot of fun! Enjoy!!!</span>
</body></html>
```



Hello, **class!** Welcome to the world of Python program. Hopefully, this will be a fasinating experience. The most important part of learing a programming language is practice. *Hope you can practice a lot and have a lot of fun! Enjoy!!!*

Create Local Pages with Python

Version 3 (helloWeb3.py): The web page content is based on a template and user input.
Compose/modify the web page template in Kompozer 's Design window, which is easier.


Hello [file:/.../helloTemplate.html] - KompoZer
File Edit View Insert Format Table Tools Help
New Open Save Publish Browse Undo Redo Anchor Link Image Table Form HTML CSS
Body Text
Variable Width
Site Manager
View: All files
Name
FTP console
Hello
697px
100px
DOM
Design
<body> <a>
Organizati... Organizati... Hello Hello
File Edit View Favorites Tools Help
Google Search More Sign In
Suggested Sites Web Slice Gallery Dic Idioms Blackboard

Hello, **%s!** Welcome to the world of Python program. Hopefully, this will be a fasinating experience. The most important part of learing a programming language is practice. *Hope you can practice a lot and have a lot of fun! Enjoy!!!*

Python programming is useful and popular in industry.

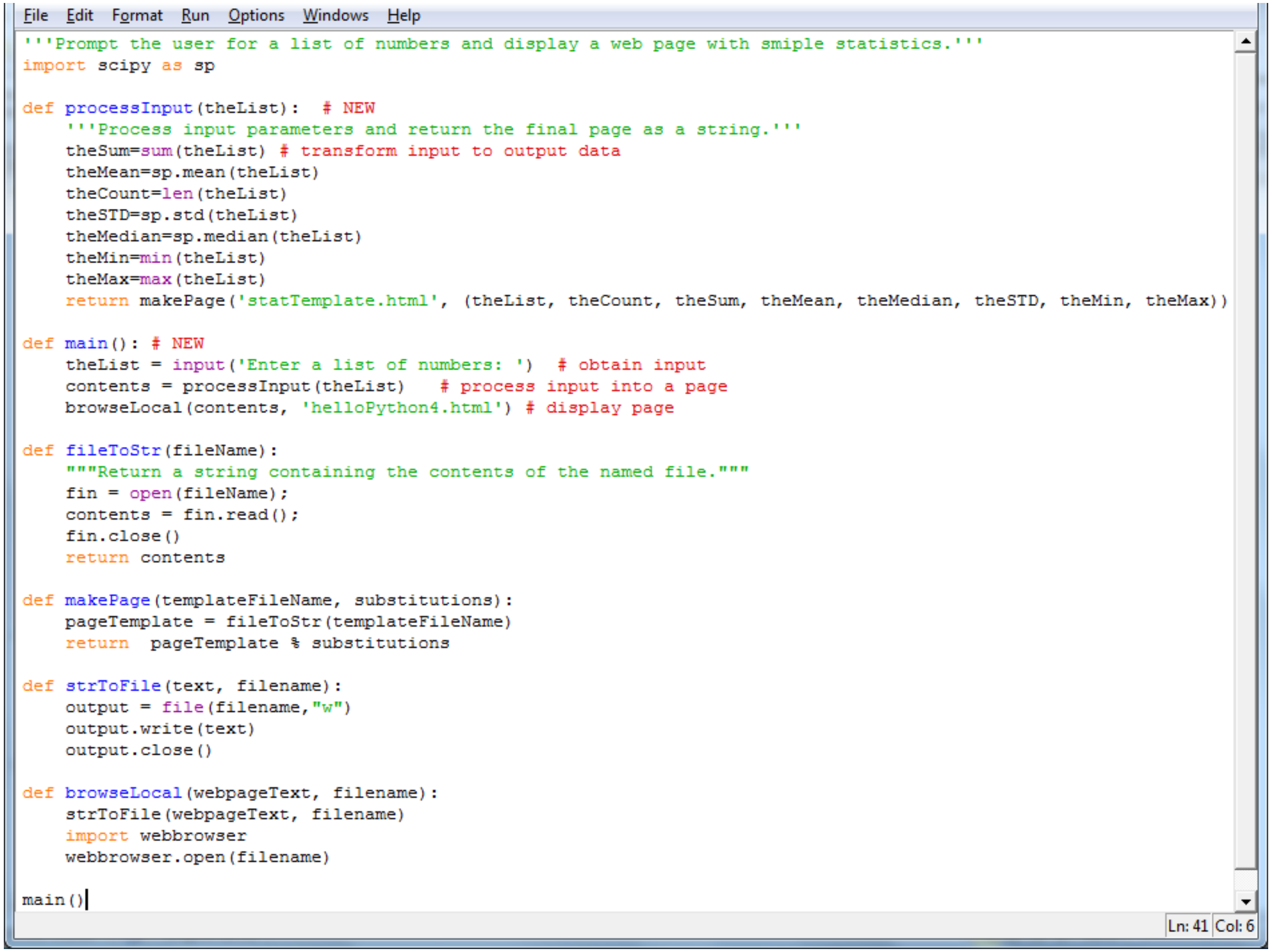
Hello, **class!** Welcome to the world of Python program. Hopefully, this will be a fasinating experience. The most important part of learing a programming language is practice. *Hope you can practice a lot and have a lot of fun! Enjoy!!!*

Python programming is useful and popular in industry.

A link to other web site

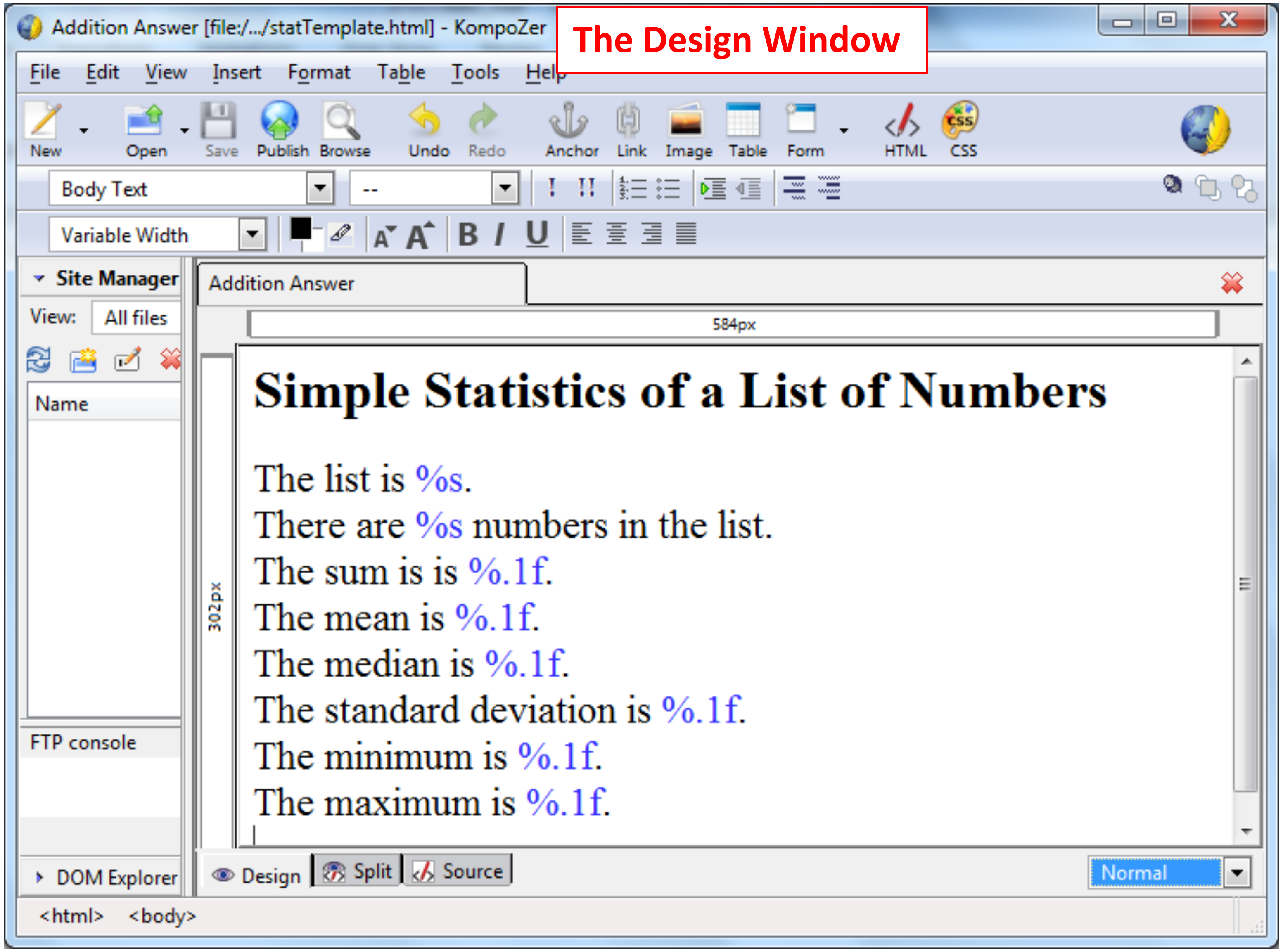
http://wiki.python.org/moin/OrganizationsUsingPython

Create a Web Page Calculating Simple Statistics of a List of Numbers with Python



```python
'''Prompt the user for a list of numbers and display a web page with smiple statistics.'''
import scipy as sp

def processInput(theList):  # NEW
    '''Process input parameters and return the final page as a string.'''
    theSum=sum(theList) # transform input to output data
    theMean=sp.mean(theList)
    theCount=len(theList)
    theSTD=sp.std(theList)
    theMedian=sp.median(theList)
    theMin=min(theList)
    theMax=max(theList)
    return makePage('statTemplate.html', (theList, theCount, theSum, theMean, theMedian, theSTD, theMin, theMax))

def main(): # NEW
    theList = input('Enter a list of numbers: ')  # obtain input
    contents = processInput(theList)   # process input into a page
    browseLocal(contents, 'helloPython4.html') # display page

def fileToStr(fileName):
    """Return a string containing the contents of the named file."""
    fin = open(fileName);
    contents = fin.read();
    fin.close()
    return contents

def makePage(templateFileName, substitutions):
    pageTemplate = fileToStr(templateFileName)
    return  pageTemplate % substitutions

def strToFile(text, filename):
    output = file(filename,"w")
    output.write(text)
    output.close()

def browseLocal(webpageText, filename):
    strToFile(webpageText, filename)
    import webbrowser
    webbrowser.open(filename)

main()
```

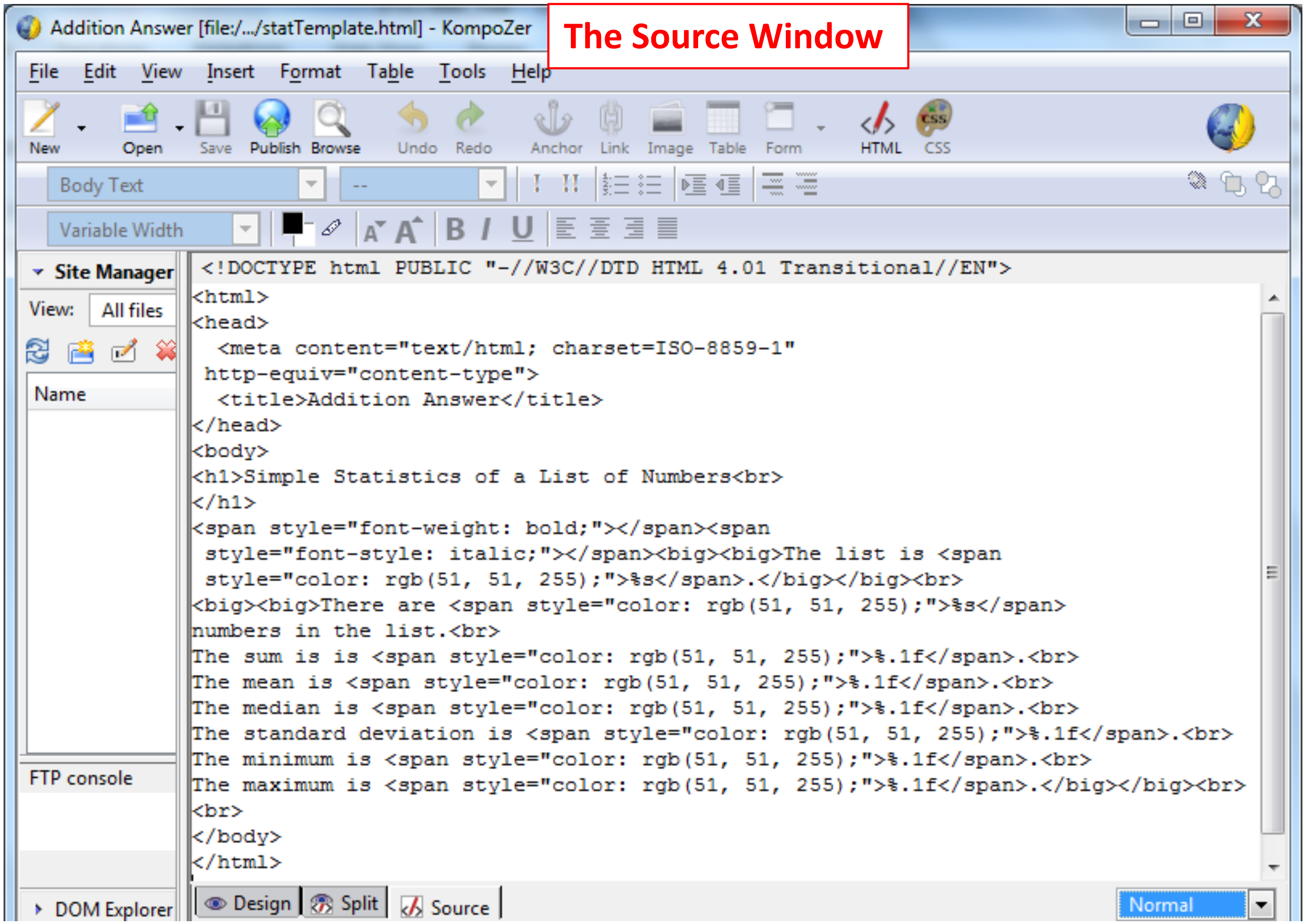
Create a Web Page Calculating Simple Statistics of a List of Numbers with Python


Addition Answer [file:/.../statTemplate.html] - KompoZer
The Design Window
File Edit View Insert Format Table Tools Help
New Open Save Publish Browse Undo Redo Anchor Link Image Table Form HTML CSS
Body Text
Variable Width
Site Manager
View: All files
Name
FTP console
DOM Explorer
Addition Answer
584px
302px
Design Split Source
Normal
<html> <body>

Simple Statistics of a List of Numbers

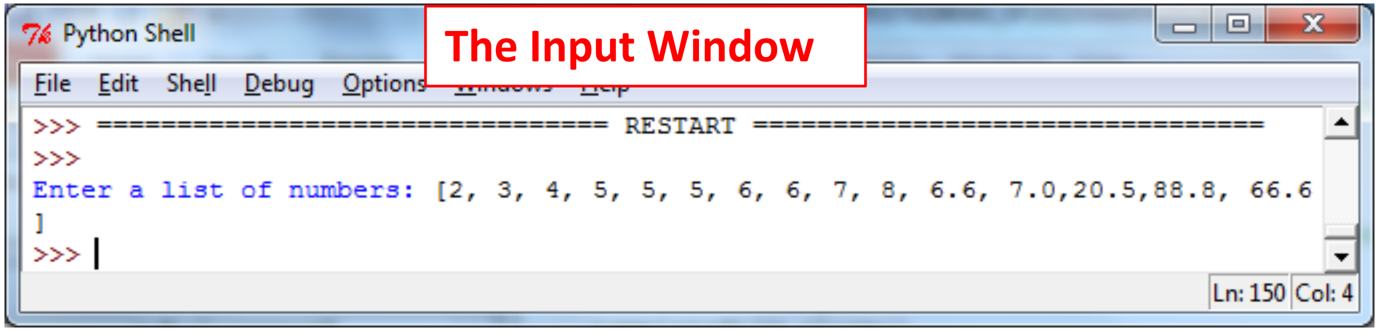
The list is %s.
There are %s numbers in the list.
The sum is is %.1f.
The mean is %.1f.
The median is %.1f.
The standard deviation is %.1f.
The minimum is %.1f.
The maximum is %.1f.

Create a Web Page Calculating Simple Statistics of a List of Numbers with Python

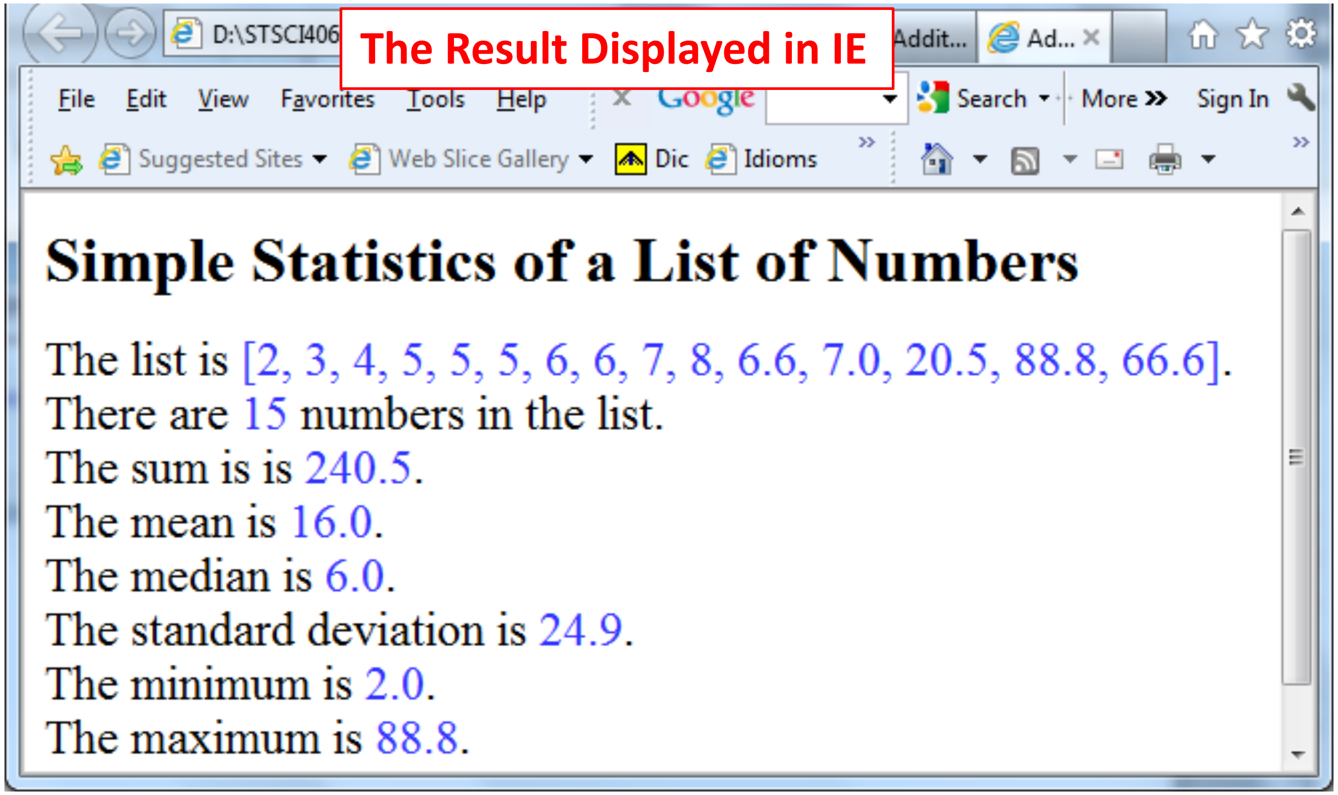

Addition Answer [file:/.../statTemplate.html] - KompoZer
The Source Window
File Edit View Insert Format Table Tools Help
New Open Save Publish Browse Undo Redo Anchor Link Image Table Form HTML CSS
Body Text
Variable Width
Site Manager
View: All files
Name
FTP console
DOM Explorer
Design Split Source
Normal

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type">
  <title>Addition Answer</title>
</head>
<body>
<h1>Simple Statistics of a List of Numbers<br>
</h1>
<span style="font-weight: bold;"></span><span
 style="font-style: italic;"></span><big><big>The list is <span
 style="color: rgb(51, 51, 255);">%s</span>.</big></big><br>
<big><big>There are <span style="color: rgb(51, 51, 255);">%s</span>
numbers in the list.<br>
The sum is is <span style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The mean is <span style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The median is <span style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The standard deviation is <span style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The minimum is <span style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The maximum is <span style="color: rgb(51, 51, 255);">%.1f</span>.</big></big><br>
<br>
</body>
</html>
```

Create a Web Page Calculating Simple Statistics of a List of Numbers with Python



```
>>> ================================ RESTART ================================
>>>
Enter a list of numbers: [2, 3, 4, 5, 5, 5, 6, 6, 7, 8, 6.6, 7.0,20.5,88.8, 66.6
]
>>> 
```



Simple Statistics of a List of Numbers

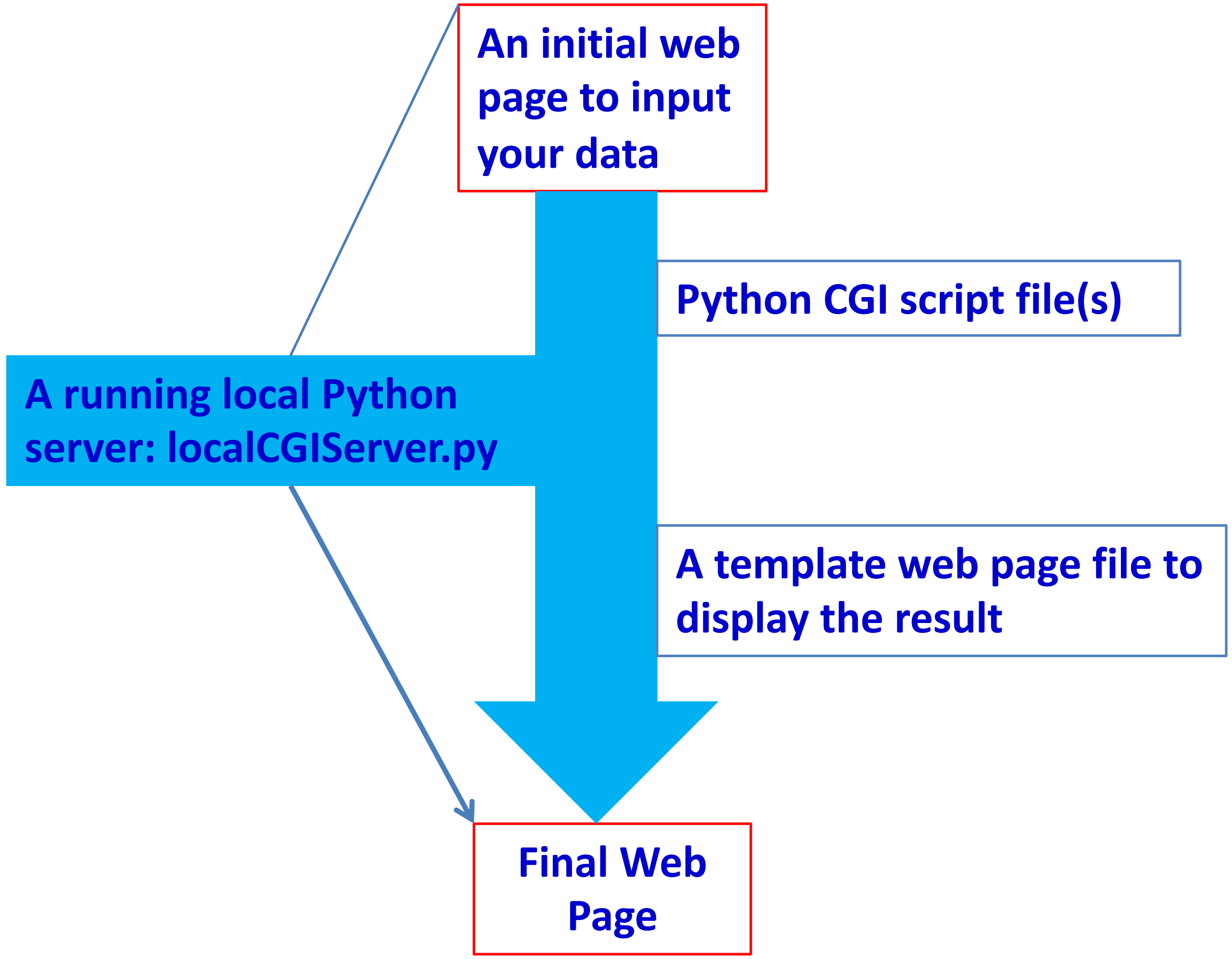
The list is [2, 3, 4, 5, 5, 5, 6, 6, 7, 8, 6.6, 7.0, 20.5, 88.8, 66.6].
There are 15 numbers in the list.
The sum is is 240.5.
The mean is 16.0.
The median is 6.0.
The standard deviation is 24.9.
The minimum is 2.0.
The maximum is 88.8.

Create Dynamic Web Pages with Python

CGI (common gateway interface) is an interface used by web servers to process information requests supplied by a browser. All the server programs end in a "**.cgi**" extension, which are all Python programs in our case; these programs are often called scripts, or **Python CGI scripts**.

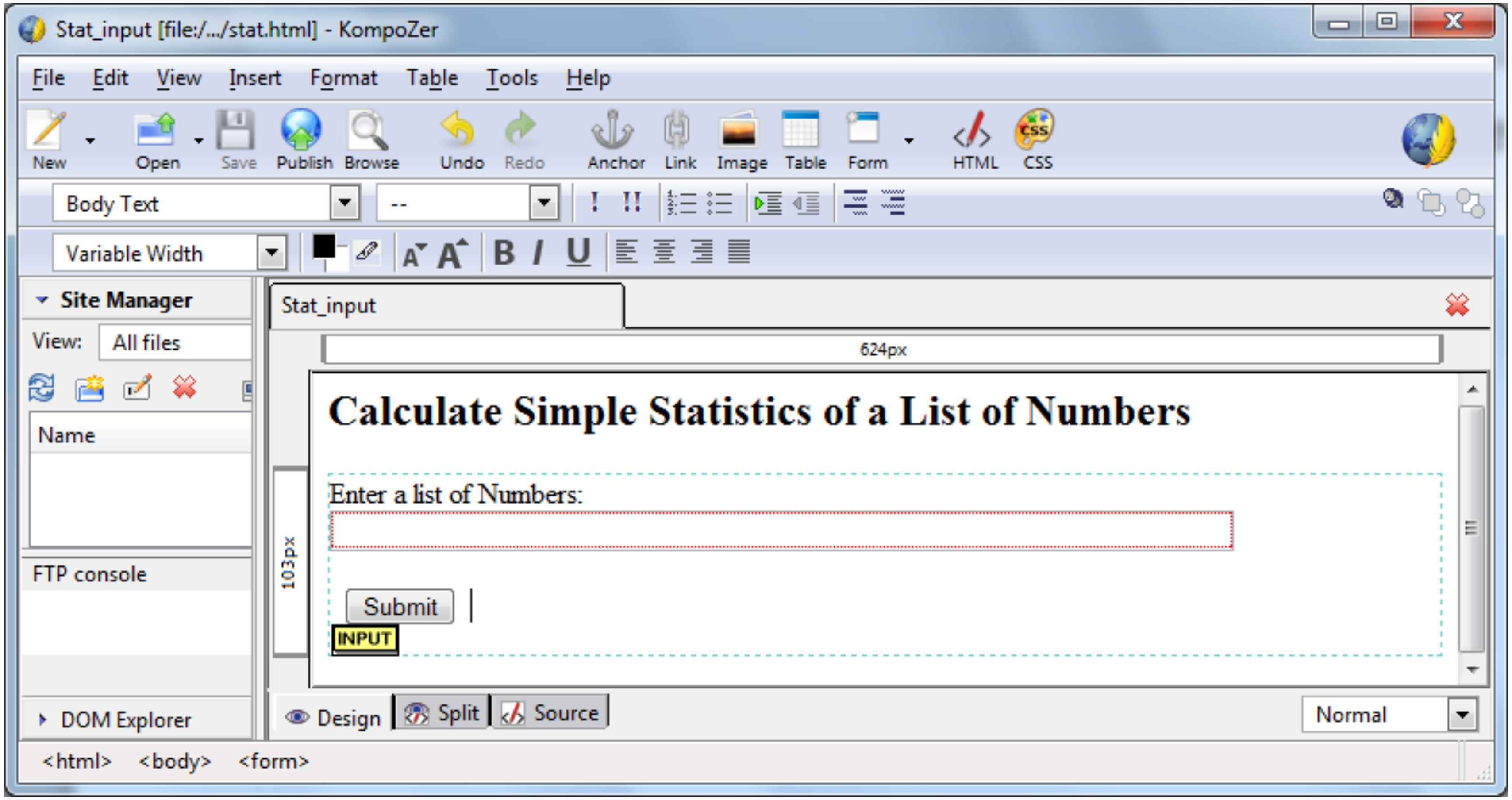
- Download the localCGIServer.py file from the course web site (within the "Software" folder)
- Save this file to the folder where you store all your web page files.
- Make sure there are **no spaces (or blanks)** in different levels of directory names.
- Start your local web server by double clicking on localCGIServer.py (in Windows OS do not start from within IDEL, etc.) and keep it running by leaving the console window open.
- Input a local link to your html file, e.g., http://localhost:8081/hello.html, in your browser's URL window.
- Now the web address is your localhost:8081, which references the local Python server we just started.
- Look at the web server console window, where you should see some activity.
- If you close the console window, you get an error when you try to reload the page.

Dynamic Local Web Page Components: How does it work?


An initial web page to input your data
Python CGI script file(s)
A running local Python server: localCGIServer.py
A template web page file to display the result
Final Web Page

Create a Dynamic Local Web Page for Calculating Simple Statistics of a List of Numbers with Python

An initial web page to input your data: the Design window


Stat_input [file:/.../stat.html] - KompoZer
File Edit View Insert Format Table Tools Help
New Open Save Publish Browse Undo Redo Anchor Link Image Table Form HTML CSS
Body Text
Variable Width
Site Manager
View: All files
Name
FTP console
DOM Explorer
Stat_input
624px
103px
INPUT
Design Split Source
Normal
<html> <body> <form>

Calculate Simple Statistics of a List of Numbers

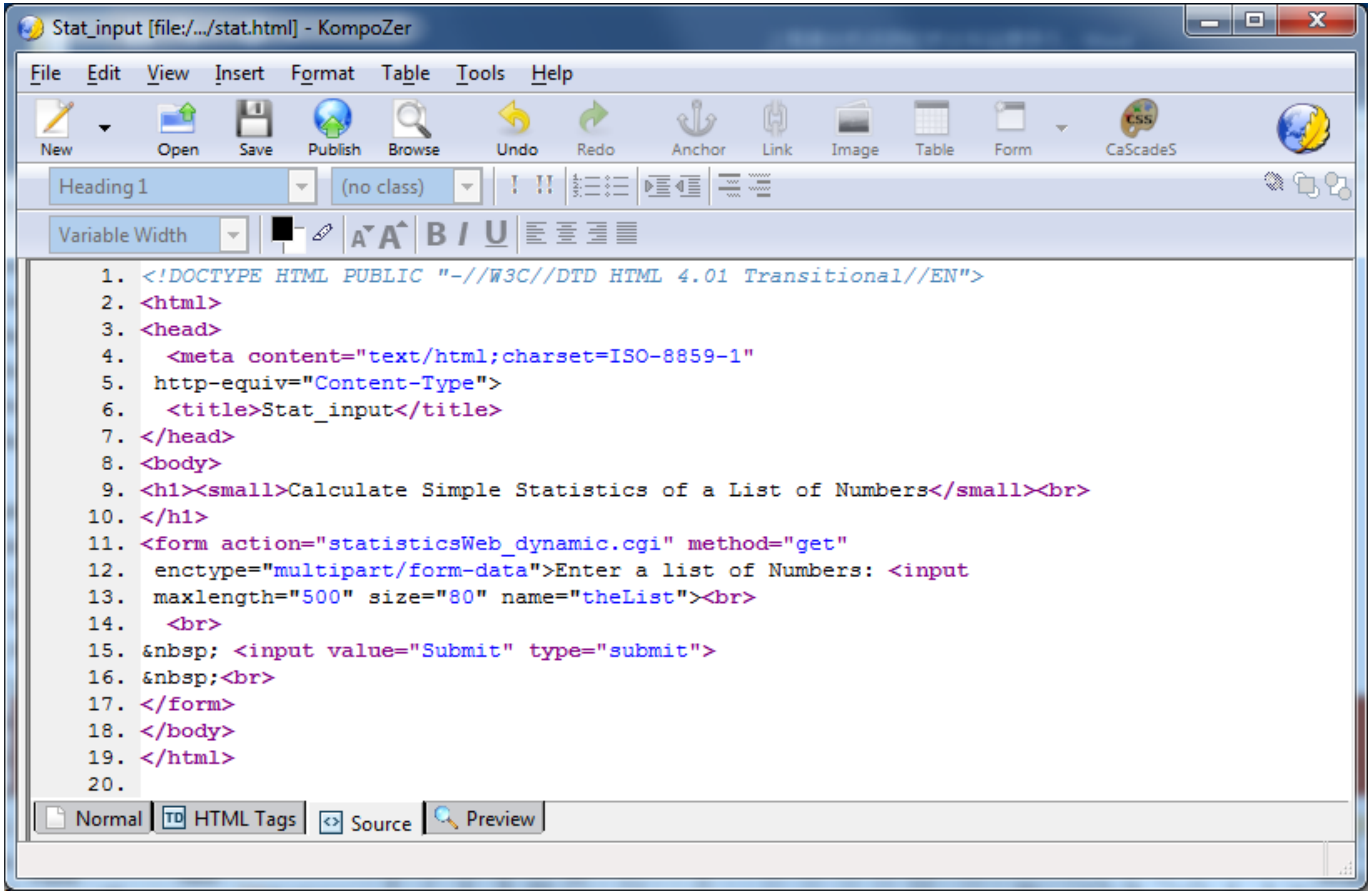
Enter a list of Numbers:

Submit

(This design can be modified from a file called commonFormFields.html, available from course web site under "Software.")

Create a Dynamic Local Web Page for Calculating Simple Statistics of a List of Numbers with Python

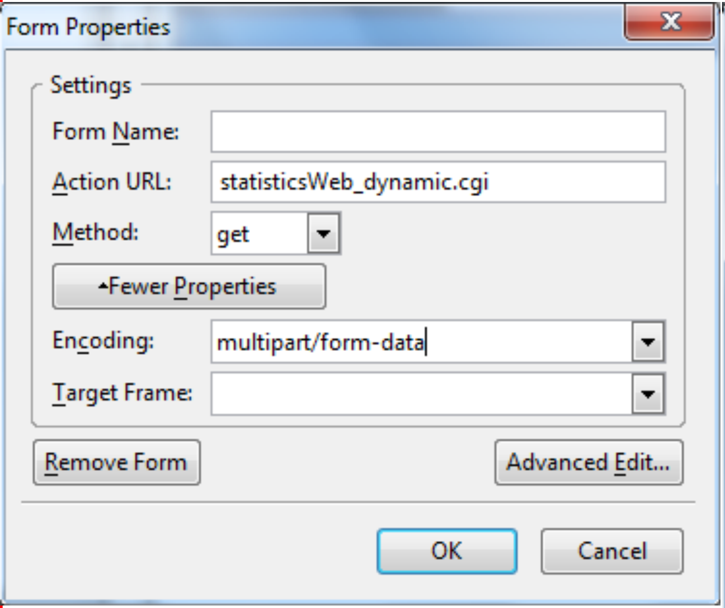
An initial web page to input your data: the Source window


Stat_input [file:/.../stat.html] - KompoZer
File Edit View Insert Format Table Tools Help
New Open Save Publish Browse Undo Redo Anchor Link Image Table Form CaScadeS
Heading 1
(no class)
Variable Width
Normal HTML Tags Source Preview

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html;charset=ISO-8859-1"
 http-equiv="Content-Type">
  <title>Stat_input</title>
</head>
<body>
<h1><small>Calculate Simple Statistics of a List of Numbers</small><br>
</h1>
<form action="statisticsWeb_dynamic.cgi" method="get"
 enctype="multipart/form-data">Enter a list of Numbers: <input
 maxlength="500" size="80" name="theList"><br>
  <br>
&nbsp; <input value="Submit" type="submit">
&nbsp;<br>
</form>
</body>
</html>

```

Create a Dynamic Local Web Page for Calculating Simple Statistics of a List of Numbers with Python

Set form/field properties and link the web page to a Python CGI script


Form Properties

Settings

Form Name:

Action URL: statisticsWeb_dynamic.cgi

Method: get

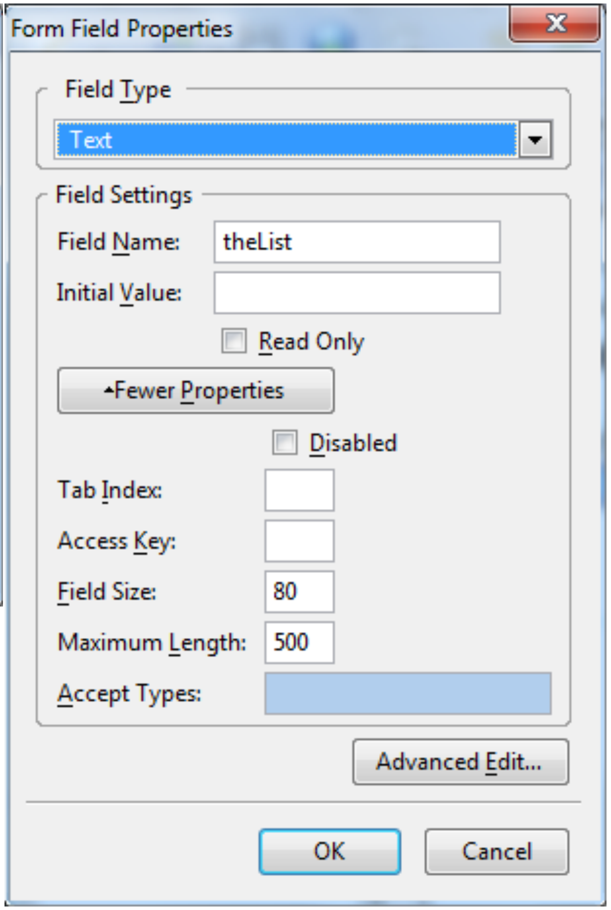
Fewer Properties

Encoding: multipart/form-data

Target Frame:

Remove Form   Advanced Edit...

OK   Cancel


Form Field Properties

Field Type: Text

Field Settings

Field Name: theList

Initial Value:

Read Only

Fewer Properties

Disabled

Tab Index:

Access Key:

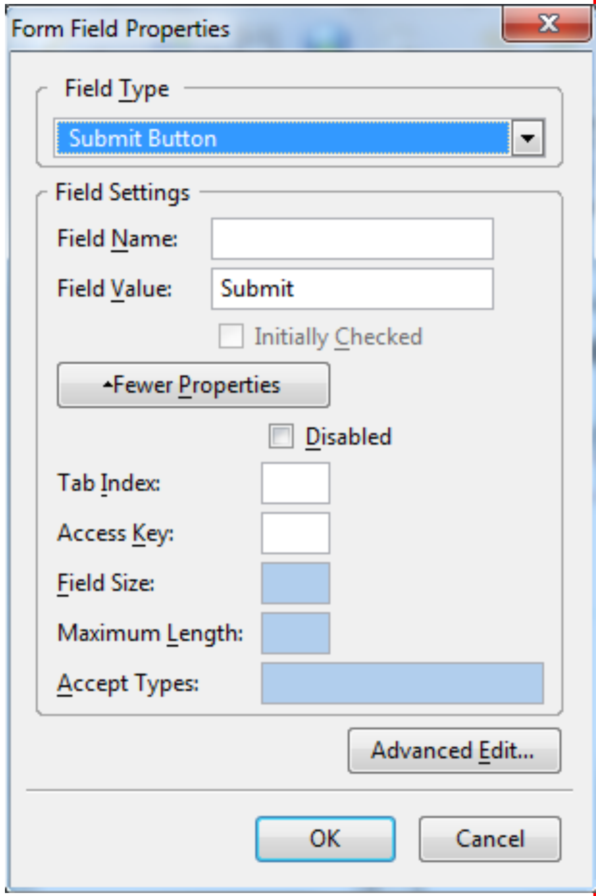
Field Size: 80

Maximum Length: 500

Accept Types:

Advanced Edit...

OK   Cancel


Form Field Properties

Field Type: Submit Button

Field Settings

Field Name:

Field Value: Submit

Initially Checked

Fewer Properties

Disabled

Tab Index:

Access Key:

Field Size:

Maximum Length:

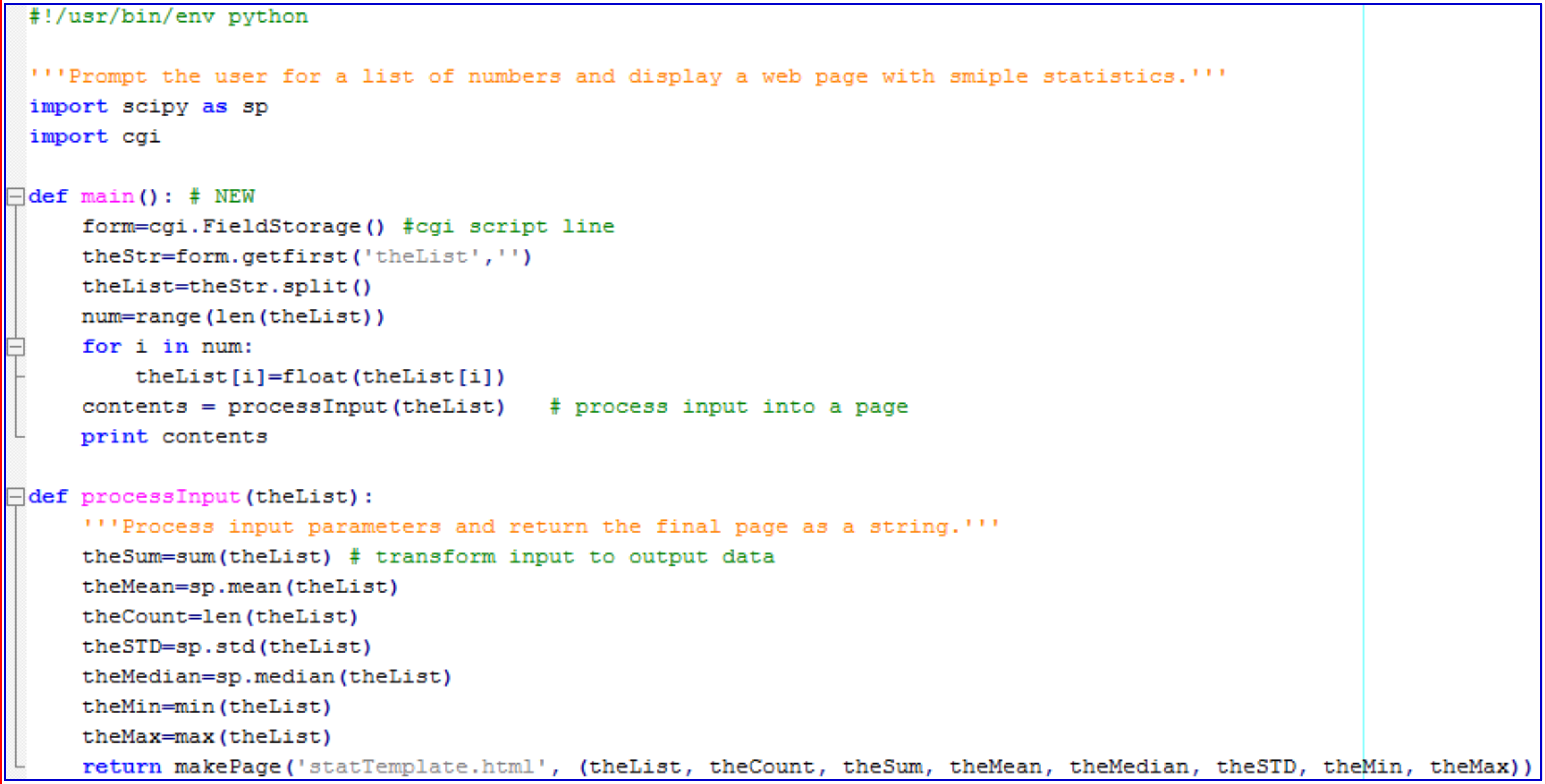
Accept Types:

Advanced Edit...

OK   Cancel

Create a Dynamic Local Web Page for Calculating Simple Statistics of a List of Numbers with Python

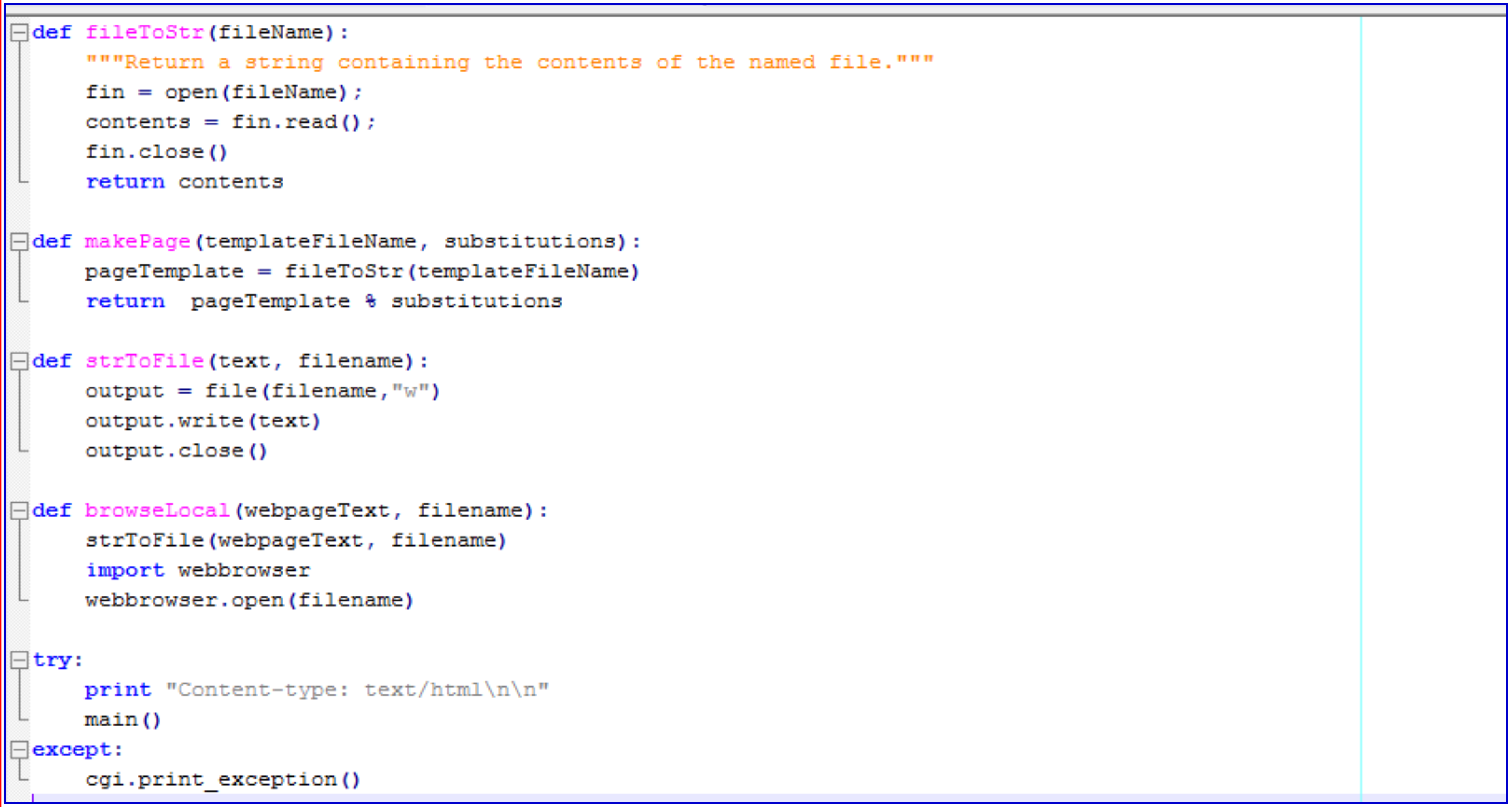
Python CGI script file(s): Part 1



```
#!/usr/bin/env python

'''Prompt the user for a list of numbers and display a web page with smiple statistics.'''
import scipy as sp
import cgi

def main(): # NEW
    form=cgi.FieldStorage() #cgi script line
    theStr=form.getfirst('theList','')
    theList=theStr.split()
    num=range(len(theList))
    for i in num:
        theList[i]=float(theList[i])
    contents = processInput(theList)   # process input into a page
    print contents

def processInput(theList):
    '''Process input parameters and return the final page as a string.'''
    theSum=sum(theList) # transform input to output data
    theMean=sp.mean(theList)
    theCount=len(theList)
    theSTD=sp.std(theList)
    theMedian=sp.median(theList)
    theMin=min(theList)
    theMax=max(theList)
    return makePage('statTemplate.html', (theList, theCount, theSum, theMean, theMedian, theSTD, theMin, theMax))
```

Create a Dynamic Local Web Page for Calculating Simple Statistics of a List of Numbers with Python

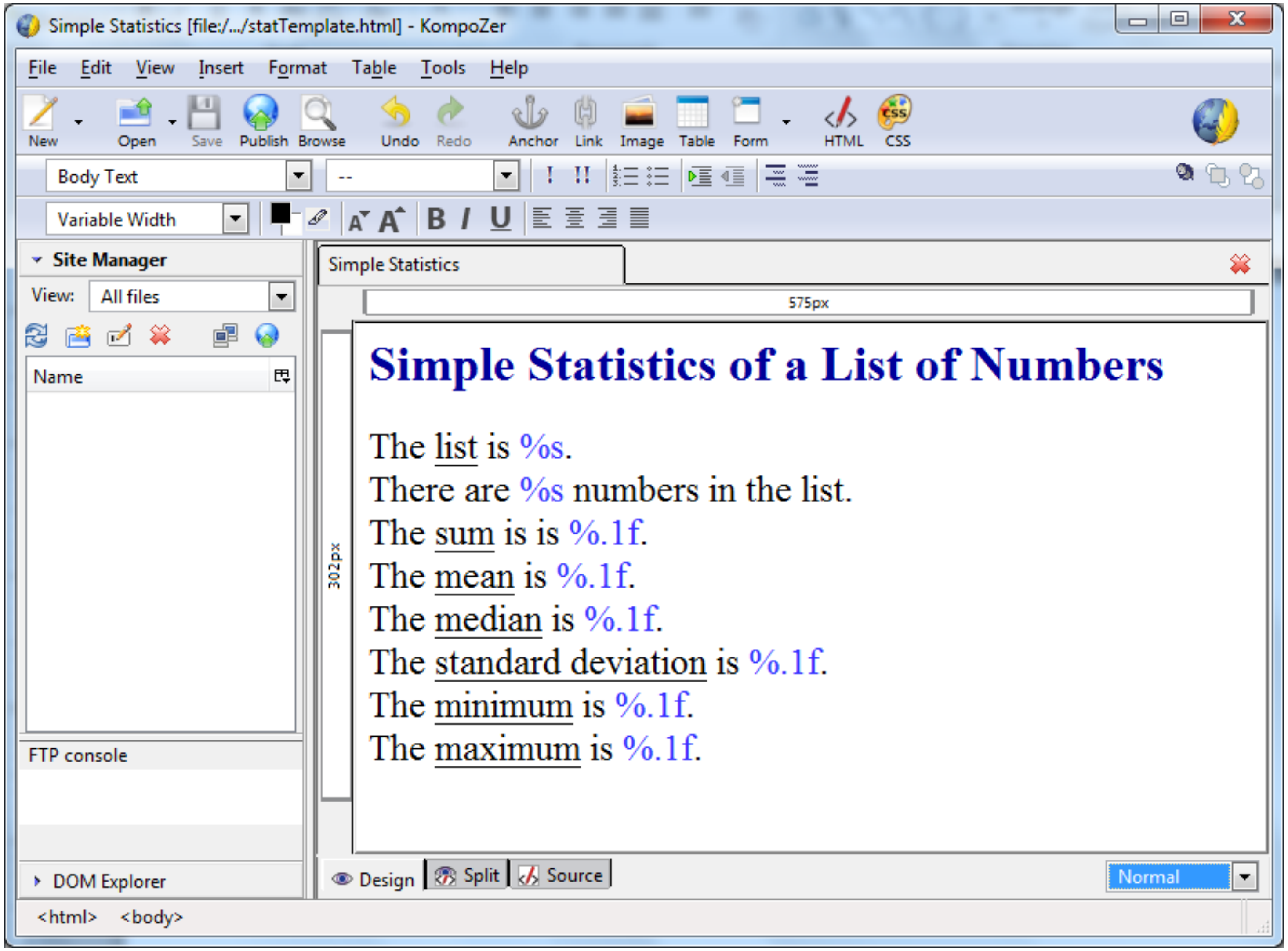
Python CGI script file(s): Part 2



```
def fileToStr(fileName):
    """Return a string containing the contents of the named file."""
    fin = open(fileName);
    contents = fin.read();
    fin.close()
    return contents

def makePage(templateFileName, substitutions):
    pageTemplate = fileToStr(templateFileName)
    return  pageTemplate % substitutions

def strToFile(text, filename):
    output = file(filename,"w")
    output.write(text)
    output.close()

def browseLocal(webpageText, filename):
    strToFile(webpageText, filename)
    import webbrowser
    webbrowser.open(filename)

try:
    print "Content-type: text/html\n\n"
    main()
except:
    cgi.print_exception()
```

Create a Dynamic Local Web Page for Calculating Simple Statistics of a List of Numbers with Python

A template web page file to display the result: the Design Window

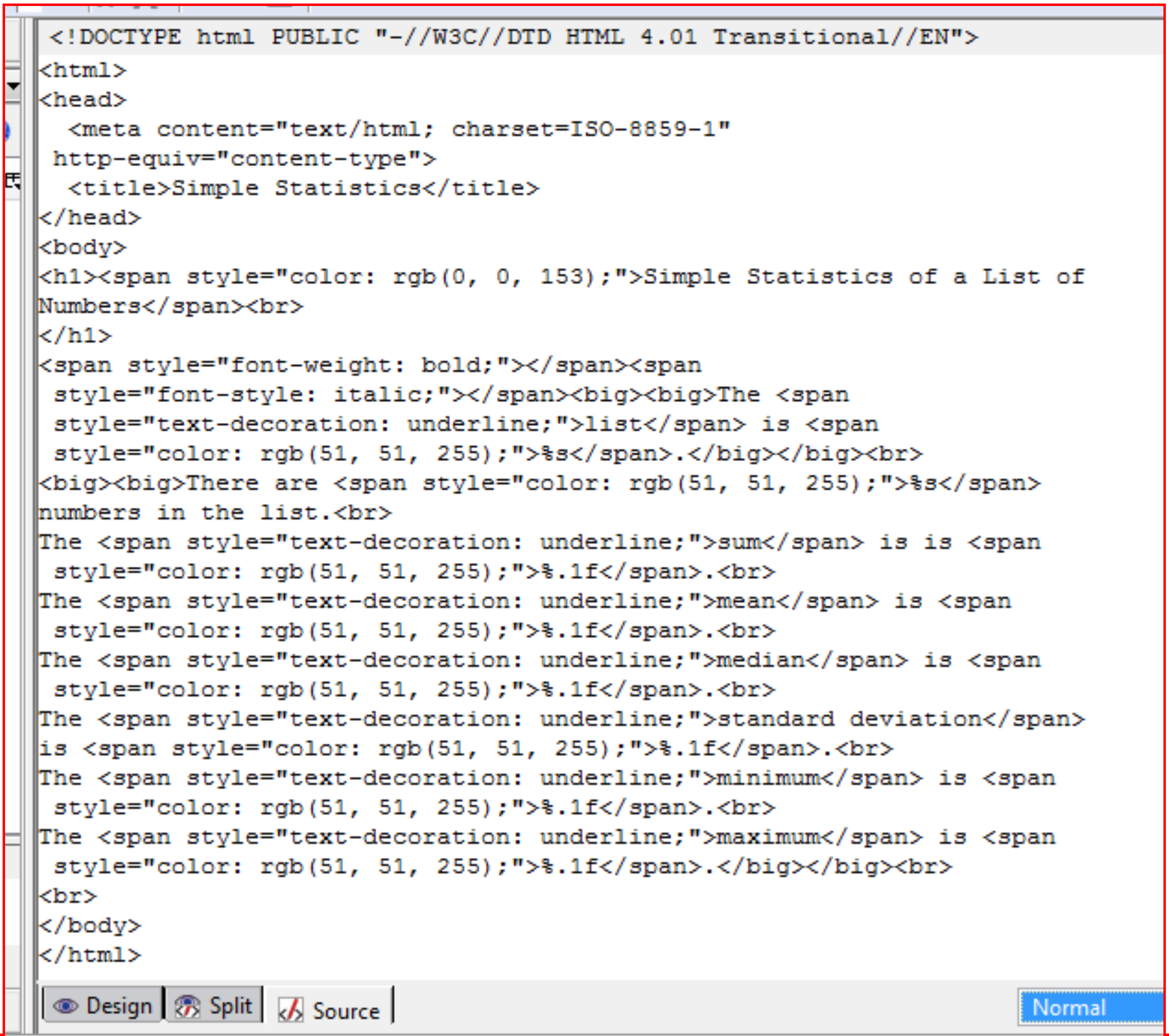

Simple Statistics [file:/.../statTemplate.html] - KompoZer
File Edit View Insert Format Table Tools Help
New Open Save Publish Browse Undo Redo Anchor Link Image Table Form HTML CSS
Body Text
Variable Width
Site Manager
View: All files
Name
FTP console
DOM Explorer
Simple Statistics
575px
302px
Design Split Source
Normal
<html> <body>

Simple Statistics of a List of Numbers

The list is %s.
There are %s numbers in the list.
The sum is is %.1f.
The mean is %.1f.
The median is %.1f.
The standard deviation is %.1f.
The minimum is %.1f.
The maximum is %.1f.

Create a Dynamic Local Web Page for Calculating Simple Statistics of a List of Numbers with Python

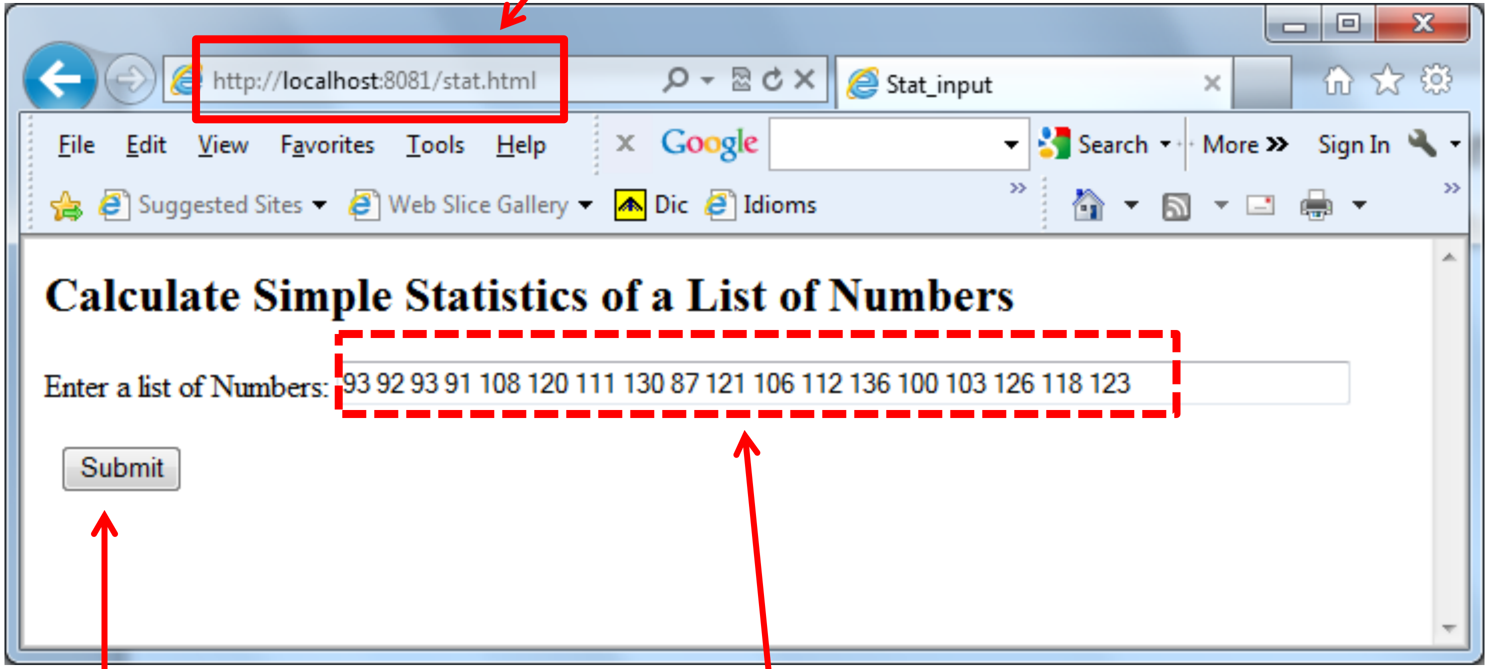
A template web page file to display the result: the Source Window



```
 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type">
  <title>Simple Statistics</title>
</head>
<body>
<h1><span style="color: rgb(0, 0, 153);">Simple Statistics of a List of
Numbers</span><br>
</h1>
<span style="font-weight: bold;"></span><span
 style="font-style: italic;"></span><big><big>The <span
 style="text-decoration: underline;">list</span> is <span
 style="color: rgb(51, 51, 255);">%s</span>.</big></big><br>
<big><big>There are <span style="color: rgb(51, 51, 255);">%s</span>
numbers in the list.<br>
The <span style="text-decoration: underline;">sum</span> is is <span
 style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The <span style="text-decoration: underline;">mean</span> is <span
 style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The <span style="text-decoration: underline;">median</span> is <span
 style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The <span style="text-decoration: underline;">standard deviation</span>
is <span style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The <span style="text-decoration: underline;">minimum</span> is <span
 style="color: rgb(51, 51, 255);">%.1f</span>.<br>
The <span style="text-decoration: underline;">maximum</span> is <span
 style="color: rgb(51, 51, 255);">%.1f</span>.</big></big><br>
<br>
</body>
</html>
```

Create a Dynamic Local Web Page for Calculating Simple Statistics of a List of Numbers with Python

1. To start the program



Calculate Simple Statistics of a List of Numbers

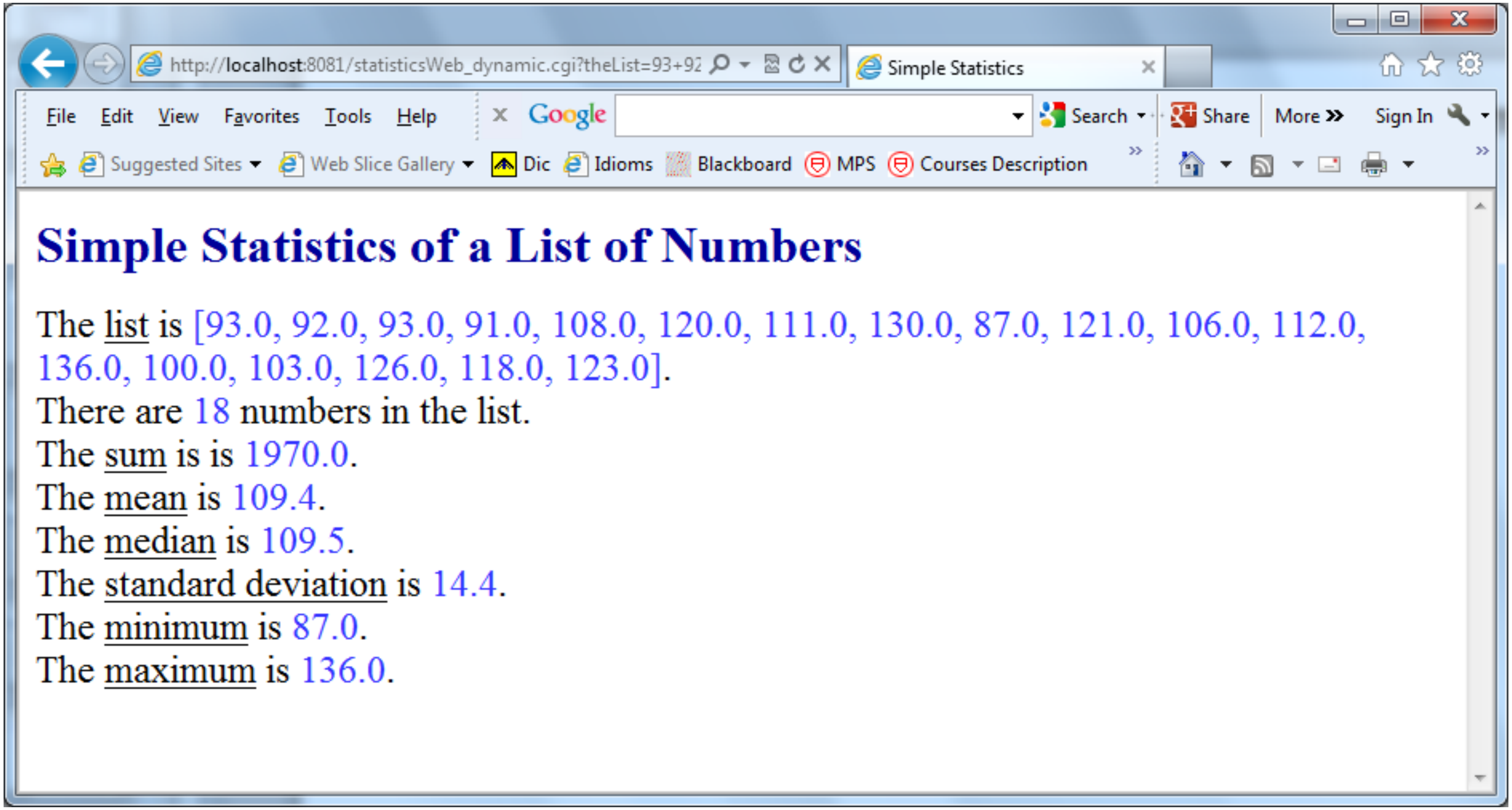
Enter a list of Numbers: 93 92 93 91 108 120 111 130 87 121 106 112 136 100 103 126 118 123

2. Enter some numbers

3. Click the button

Create a Dynamic Local Web Page for Calculating Simple Statistics of a List of Numbers with Python

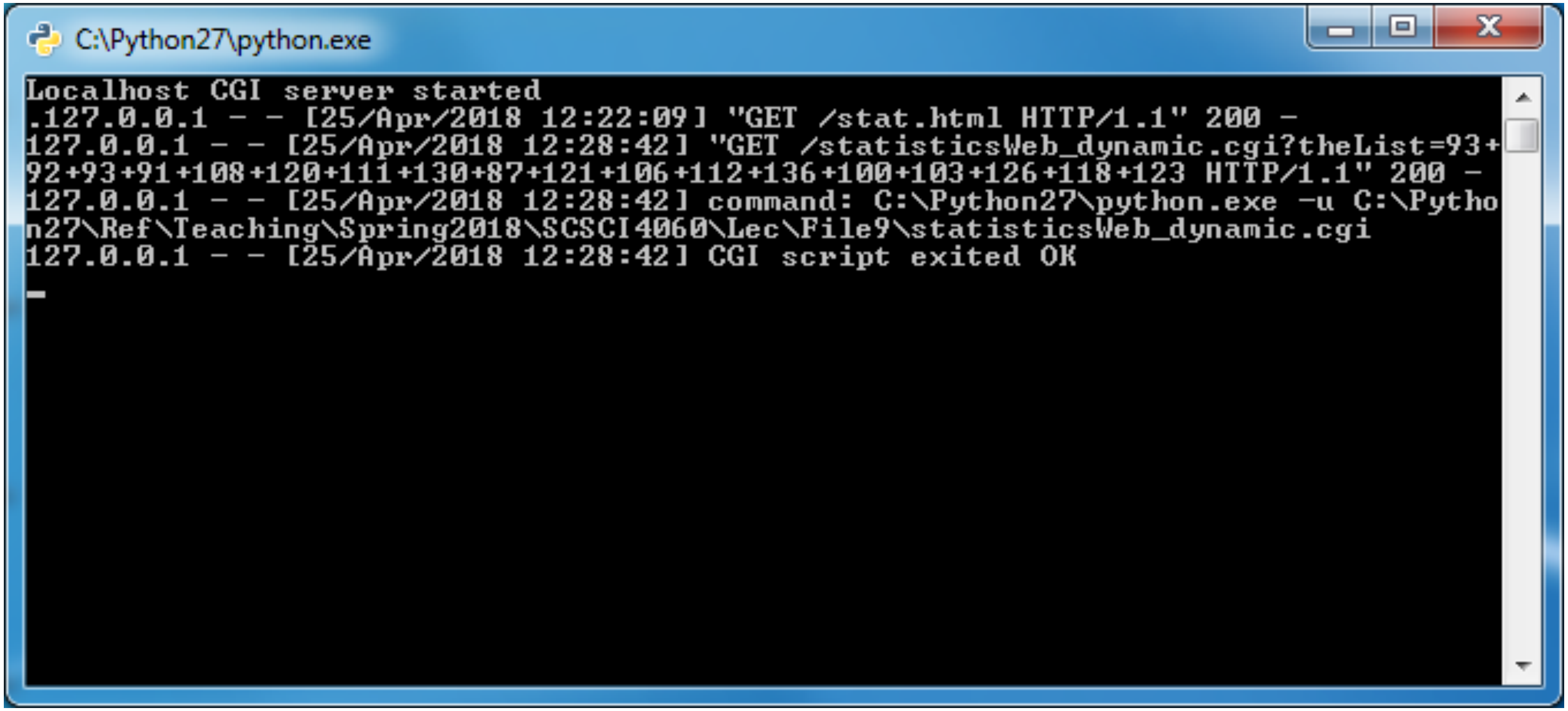
The Final Web Page



Simple Statistics of a List of Numbers

The list is [93.0, 92.0, 93.0, 91.0, 108.0, 120.0, 111.0, 130.0, 87.0, 121.0, 106.0, 112.0, 136.0, 100.0, 103.0, 126.0, 118.0, 123.0].
There are 18 numbers in the list.
The sum is is 1970.0.
The mean is 109.4.
The median is 109.5.
The standard deviation is 14.4.
The minimum is 87.0.
The maximum is 136.0.

Local CGI Server Log



```
Localhost CGI server started
.127.0.0.1 - - [25/Apr/2018 12:22:09] "GET /stat.html HTTP/1.1" 200 -
127.0.0.1 - - [25/Apr/2018 12:28:42] "GET /statisticsWeb_dynamic.cgi?theList=93+
92+93+91+108+120+111+130+87+121+106+112+136+100+103+126+118+123 HTTP/1.1" 200 -
127.0.0.1 - - [25/Apr/2018 12:28:42] command: C:\Python27\python.exe -u C:\Pytho
n27\Ref\Teaching\Spring2018\SCSCI4060\Lec\File9\statisticsWeb_dynamic.cgi
127.0.0.1 - - [25/Apr/2018 12:28:42] CGI script exited OK
_
```