Today's Agenda

- Assignment 2: Q/A
- Unit 7 – Intro to programming & Python
- Bonus exercise explained
- If time permits, midterm solutions - Review #3

CMPT 165
Unit 7 – Intro to Programming Part 2

July 17th, 2015

Q: what are key components to good webpages?

1. Good **content**
- Readable pages
- High-quality (check grammar, typos, etc.)
2. Well-**structure**d (markup)
- Validated code → ensures render as expected in "not-as-intelligent" devices
- Think accessibility issues (e.g. have you provided attributes like **`alt`**, **`abbr`**?)
3. Well-**style**d (CSS)
- Think visual design principles: Colour schemes for contrast? Margins aligned?...
- Usability issues: Does chosen colour scheme work? Layout consistent?
4. Adequate **user-interaction**
- Amuse your visitors
- Provide proper feedback
    - Simple in markup/CSS:
        - Tooltips in **`<img>`** | **`<a>`** | **`<abbr>`** tags (some via **`title`** attribute)
        - Pseudo-class **`:hover`** | **`:active`** **(style is changed in response to mouse)**
    - Elaborate: Python programming

Programming

What?

- Task of creating a program
- What is a program?
  - List of *instructions* a software follows to perform a task
    - Instructions… language spoken to computer

Why?

- Do lot's of cool things..
  - Automate (complex) calculations, i.e. 111999991900522*101010889991
  - In this course, allows us to generate dynamic markup

How?

- Via an **interface** (bridge/exchange between X and Y):
  - Text-based: "command-line"
  - Graphical: "GUI"
    - For Python, use IDLE

Interface: a.k.a. "shell"

GUI: Graphical User Interface

X: program developer = programmer (i.e. you)
Y: computer

Buzz words so far…

- Program
- Dynamic HTML (markup)

- Developer

- Interface
- GUI
- Shell

Programming

What?

- Task of creating a program
- What is a program?
  - List of *instructions* a software follows to perform a task
    - Instructions… language spoken to computer

Q: What other languages can you learn in CMPT course?

A: C, C++, Java, C#, … many!

Q: Which ones are relevant to webpage dev.?

A: Javascript, Python, …

Q: Why Python?

Low-level language (hard to learn) ☹

High-level language (easy to learn) ☺

Why Python?

- Relatively easy to learn
- A *general-purpose* programming language
  - i.e. can do lots of things:
    - Systems programming
    - Database
    - Fast prototyping
    - Scientific computing (research)
    - Web programming ← We'll focus on this in CMPT 165
- Free
- "Portable": cross-platforms, i.e. Windows, Mac, Linux,…
- Lot's of built-in tools (you can use other's sophisticated code)

Languages you'll have learned…

- Markup: XHTML 1.0 (HTML5)
- Styling language: CSS (levels 1,2,3)
- Programming language: Python

BTW, you've also touched on Extensible Markup Language (XML)!?
…
Remember when?
Ans: SVG

Q: markup vs. programming?

- **Markup**: annotate a document
- **Programming**:

Input data → Process → Output data

Buzz words so far…

- Program
- Dynamic HTML

- Developer

- Interface
- GUI
- Shell

- Data
- Input/Output (I/O)
- Process

Buzz words so far…

- Program
- Dynamic HTML

- Developer

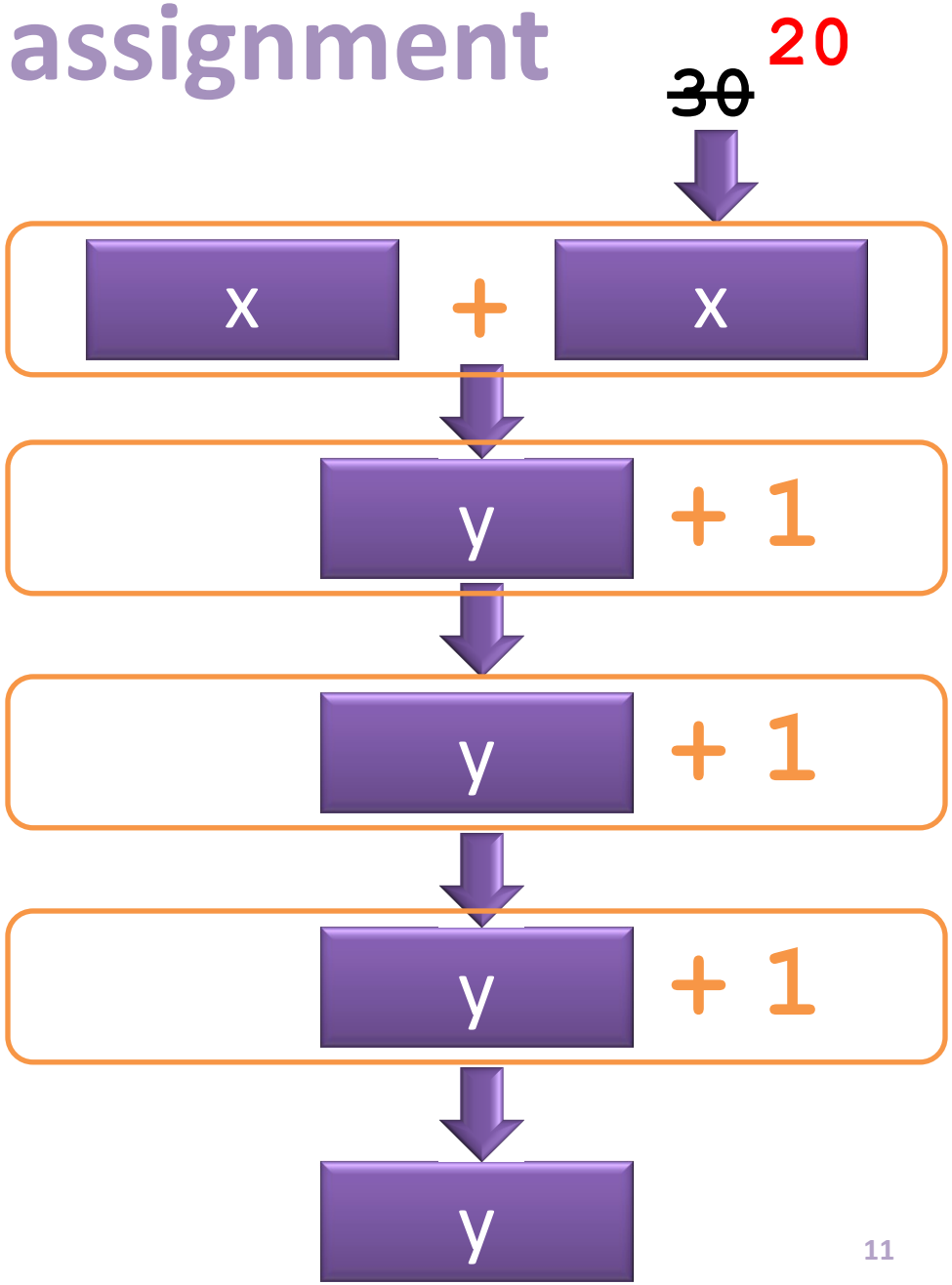
- Interface
- GUI
- Shell

- Data
- Input/Output (I/O)
- Process

- Data
  - Variable
  - Assignment
  - Types

Variable assignment

```
>>> x = 30
>>> x + x
60
>>> x = 20
>>> x + x
40
>>> y = x + x
40
>>> y = y + 1
41
>>> y = y + 1
42
>>> y += 1
43
```



Variable assignment: shorthand

NEW: Can do multiple assignments with 1 equal sign

```
>>> x,y = 2,2
>>> x
2
>>> y
2
>>> x,y,z = 2,3,11
>>> z
11
```

NEW: Can assign multiple variables to same value in one line

```
>>> a=b=c=1
>>> a
1
>>> c
1
```

Buzz words so far…

- Program
- Dynamic HTML
- Developer
- Interface
- GUI
- Shell
- Data
- Input/Output (I/O)
- Process

- Data
  - Types: Numeric, …
  - Variable
  - Assignment (shorthand)
- Operations on data
  - Arithmetic
  - Logical (next week)
  - …

Arithmetic operations

x+1	addition
x-1	subtraction
x*x	multiplication
x/2	division
x//2	integer division
	e.g. 5 // 2 returns 2
%	modulus
	e.g. 5 % 2 returns 1
x**2	exponent
	e.g. 3**2 returns 9

Arithmetic operations

```
>>> x = 25
>>> x - x
?
>>> x = 120
>>> x + x - 20
?
>>> y = x / 2
?
>>> y = y - 9
?
>>> y = y * 5
?
>>> y += y
?
```

```
>>> x = 25
>>> x - x
0
>>> x = 120
>>> x + x - 20
220
>>> y = x / 2
60
>>> y = y - 9
51
>>> y = y * 5
255
>>> y += y
510
```

Arithmetic operations

```
>>> var1=4
>>> var2=6**2+var1
?
>>> x=var1*var2
?
>>> x%=8
?
>>> z=x+var1*2
?
>>> z//=2
?
>>> z%=3
?
```

```
>>> var1=4
>>> var2=6**2+var1
40
>>> x=var1*var2
160
>>> x%=8
0
>>> z=x+var1*2
8
>>> z//=2
4
>>> z%3
1
```

Buzz words so far…

- Program
- Dynamic HTML
- Developer
- Interface
- GUI
- Shell
- Data
- Input/Output (I/O)
- Process
- Data
  - Types: Numeric, …
  - Variable
  - Assignment (shorthand)
- Operations on data
  - Arithmetic
  - Logical (next week)
  - …
- Functions

Data types

1. Numerical

- Arithmetic operations
- …

```
>>> var1=4
>>> z=1+var1*2
>>> z//=2
```

2. Characters (multiple: known as "strings")

- Possible operations:
  - Transform to upper/lower case
  - **Concatenation**
  - …

```
>>> var1='Hello '
>>> var1*=3
>>> var1
'Hello Hello Hello '

>>> var1='Hello'
>>> var2=' world'
>>> var1+var2
'Hello world'
```

Symbols are "**overloaded**"

- i.e. reused to do something else

Overloaded symbols

Example type-dependent operations:

	Numeric	Strings
`var1*=m`	Multiply `var1` by `m` and store result back in `var1`	Replicate itself `m` times

`var1+var2`	Addition
	
	e.g.
	`>>> var1=1`
	`>>> var2=1`
	`>>> var1+var2`
	`2`
		Concatenation
		
		e.g.
		`>>> var1='hello '`
		`>>> var2='world'`
		`>>> var1+var2`
		`'hello world'`

Buzz words so far…

- Program
- Dynamic HTML

- Developer

- Interface
- GUI
- Shell

- Data
- Input/Output (I/O)
- Process

- Data
  - Variable
  - Types: numeric, strings,…
  - Assignment (shorthand)

- Operations
  - Numeric: arithmetic
  - Logical (examined next week)
  - Strings: Concatenation
  - Overloaded symbols

  - Functions

function

$$f(x,y)=x^2+y^2$$

Function: a **process**

aka="also known as"

- Takes some **input** data (aka arguments), generate some **output** data
- We've seen similar notation, e.g. in CSS:

```
url(myfile.png);          /* URL of file */
rgba(100,0,0);            /* colour + opacity spec. */
hsla(100,10%,10%,0.5);    /* colour + alpha spec. */
```

- In above, it's a mathematical function
- You can use/implement a lot of other functions in Python
- There are lots of functions implemented by others (in Python libraries)…

Simple functions

- Print **statement**, used to print its **arguments** on screen

```
>>> print “Hello”
```

- Syntax changed in Python 3.x:

```
>>> print(“Hello")
```

- This function:
  1. Takes as argument (input) “Hello”
  2. Processing: none
  3. Flush the output to screen

Defining your own functions

Use the keyword **def** (**define**) & syntax:

```
>>> def name_of_function(input1,input2):
    return input1+input2
```

Example 1: calculate the square root of an input number

```
>>> def sqr(x):
    return x*x
>>> sqr(5)
25
```

Programmer's practice: square of number

Example 2: temperature conversion: Celsius (C°) to Fahrenheit degree (F°)

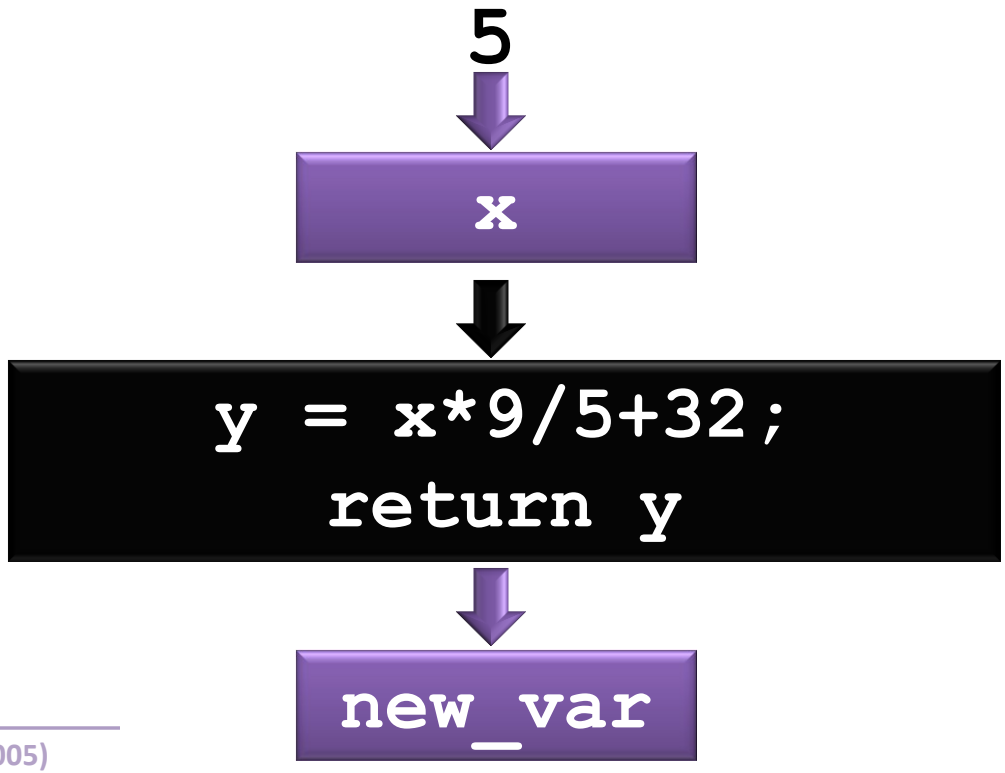
```
>>> def Celsius_to_Fahrenheit(x):
     y=x*9/5+32
     return y
>>> new_var=Celsius_to_Fahrenheit(5)
>>> new_var
41
```

What is new_var?

Function as "black box"

```
>>> def Celsius_to_Fahrenheit(x):
            y=x*9/5+32;
            return y
>>> x=5; new_var=Celsius_to_Fahrenheit(x)
>>> new_var
41
```

y=(5)*9/5+32



Quick exercise

Given this example:

```
>>> def Celsius_to_Fahrenheit(x):
    return x*9/5+32
>>> new_var=Celsius_to_Fahrenheit(1)
>>> new_var
25
```

```
          y = x*9/5 + 32
    (y - 32) = x*9/5
(y -32)*5/9 = x
```

Q: Which is input variable to your new function?

<u>Your task now</u>: write code Fahrenheit ➔ Celsius

```
>>> def Fahrenheit_to_Celsius(y):
    return (y-32)*5/9
>>> new_var=Fahrenheit_to_Celsius(50)
>>> new_var
10
```

Buzz words so far…

- Program
- Dynamic HTML

- Developer

- Interface
- GUI
- Shell

- Data
- Input/Output (I/O)
- Process

- Data
  - Variable
  - Types: numeric, strings,…
  - Assignment (shorthand)

- Operation
  - Numeric: arithmetic
  - Logical (examined next week)
  - Strings: Concatenation
  - Overloaded symbols

  - Functions
  - Statements

Statements

...An instruction to IDLE

Execute the statement when you press ENTER

```
>>> var1=4
>>> var2=var1*2
8
```

Output from IDLE in blue

>>> with green background indicates we're working in command window

S.G.: statements that are arithmetic in nature aka numeric expressions

- Can store these statements to form a program; e.g.

myfirstprogram.py

```
var1=4;
var2=var1*2;
```

Text editor of IDLE in peach color

- Statements are executed in order provided
- To **execute** all statements, press F5 key to run the saved program

Buzz words so far…

- Program
- Dynamic HTML

- Developer

- Interface
- GUI
- Shell

- Data
- Input/Output (I/O)
- Process

- Data
  - Variable
  - Types: numeric, strings,…
  - Assignment (shorthand)

- Operations
  - Numeric: arithmetic
  - Logical (examined next week)
  - Strings: Concatenation
  - Overloaded symbols

  - Functions
  - Statements

About the bonus…

CMPT 165 - Bonus Exercise 1

INSTRUCTIONS

1. If you haven't already, download and install Python version 2.7. **Note: Although the latest version is 3.x, we'll stick with 2.7 because this is the only version available on the course web server.** Of course, if you're working in an on-campus lab, it's already installed on all Windows computer (choose Start → All Programs → Python_Collections → Python 2.7).

   Select the Python → IDLE (Python GUI).

   Create an editor window (where you type your programs) by selecting File → New Window. Also, see instructions for using Python.

   Type (or copy and paste) this code into the editor window. Save it as `first.py`.

   ```
   print "I am in the IDLE program..."
   print "Running my first Python script... Life is good!"
   ```

   Press F5 (or select Run → Run Module) to run the program.

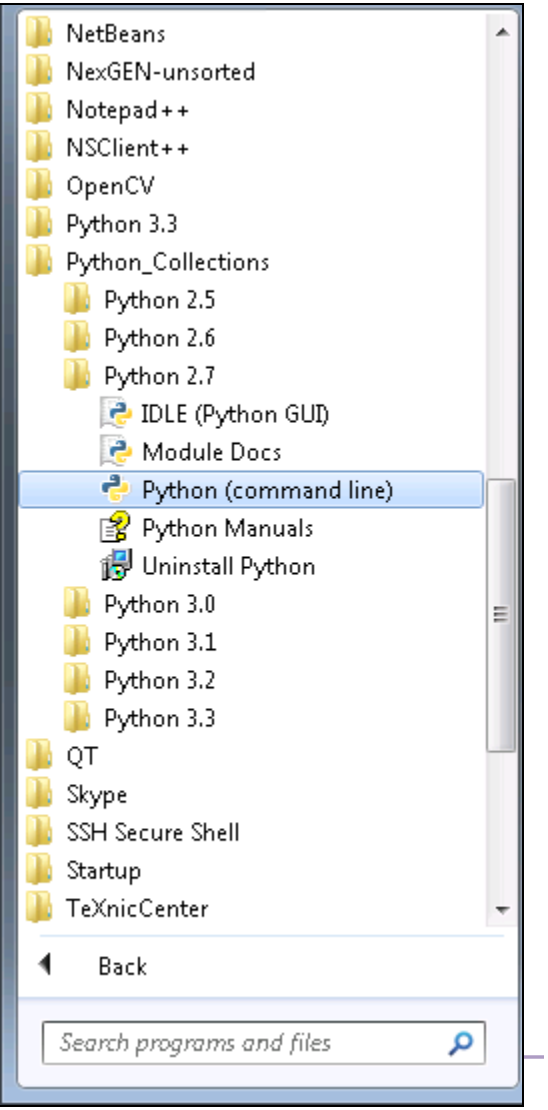
2. Modify `first.py` so it prints this instead:

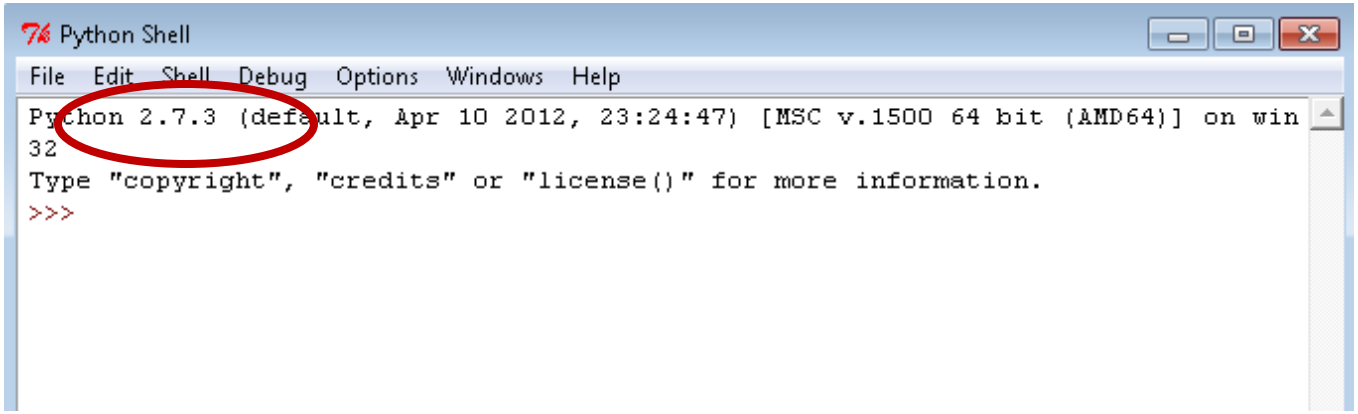
   ```
   I am a Python program...
   ...helping my author to finish a lab exercise for "CMPT 165".
   ```

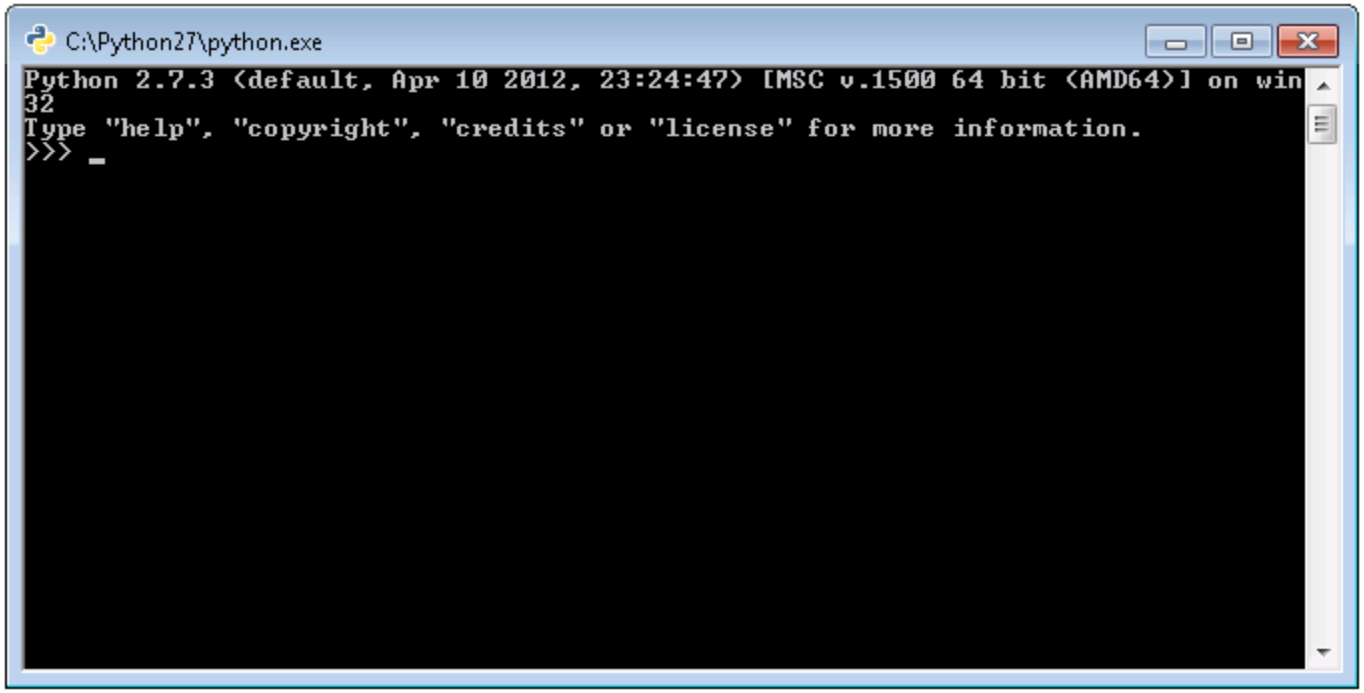
   In order to print the double quotes in the second line, you will have to use single quotes to enclose the string: `'This course is "CMPT 165".'`

3. In the same Python program, add new `print` statements to find out what these expressions evaluate to:
   i. Multiply 5012 by 2988912
   ii. Divide 2988912 by 991
   iii. Sum of these 3 numbers: 5550, 5012, 298892
   iv. Product of the following two sums:
      i. 5662 added by 56665
      ii. 5650 added by 8999
   v. Addition of the following two products:

Using Python







Steps (reviewed):

1. In the Python shell,

   open its Editor

   (File → New)

2. Type statements in Editor as instructed

3. Save as **`first.py`**

4. Execute **`first.py`** by pressing F5

first.py - Z:/165web/demos/py/first.py

File Edit Format Run Options Windows Help

```
print "I am a Python program..."
print '...helping my author to finish a lab exercise for "CMP

print 5012*2988912

print 2988912 / 991

print 5550 + 5012 + 298892
print (5662 +56665 ) * ( 5650 + 8999 )
print 51*665 + 1650*299

print 19111 % 112

print 19111//10

print 9929**120


print "9929**120"
```

Ln: 3 Col: 0

After pressing F5, the shell is updated to print the output of each statement

first.py - Z:/165web/demos/py/first.py

File Edit Format Run Options Windows Help

```
print "I am a Python program..."
print "...helping my author to finish a lab exercise for "CMP

print 5012*2988912

print 2988912 / 991

print 5550 + 5012 + 298892
print (5662 +56665 ) * ( 5650 + 8999 )
print 51*665 + 1650*299

print 19111 % 112
```

Ln: 3 Col: 0

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.3 (default, Apr 10 2012, 23:24:47) [MSC v.1500 64 bit (AMD64)] on w
in32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
I am a Python program...
...helping my author to finish a lab exercise for "CMPT 165".
14980426944
3016
309454
913028223
527265
71
1911
42526660829106213863783897313481090433714936357263351268981846838122921132158
0
20171033563655134704687702774885881562005757858911159885103664944685553643406
0
15838287572438837601917116853300929365803708023501720872667208813357160779389
4
86680582154174445951295827355865095900029046129560501564572670906972816686314
8
71105076414085484654224428155566404861621071887957693755604392901330115479657
4
31715206138918819447587868517708014326030397600404141075734825908674852291204
1
559888414401
9929**120

>>>
```

www.cs.sfu.ca/CourseCentral/165/lisat/assign/B1.html

Apps | 165web | Pt2-Requests | bus | gsheets | lts | Other bookmarks

CMPT 165 - Bonus Exercise 1

2. Modify `first.py` so it prints this instead:

```
I am a Python program...
...helping my author to finish a lab exercise for "CMPT 165".
```

   In order to print the double quotes in the second line, you will have to use single quotes to enclose the string: `'This course is "CMPT 165".'`

3. In the same Python program, add new `print` statements to find out what these expressions evaluate to:
   i. Multiply 5012 by 2988912
   ii. Divide 2988912 by 991
   iii. Sum of these 3 numbers: 5550, 5012, 298892
   iv. Product of the following two sums:
      i. 5662 added by 56665
      ii. 5650 added by 8999
   v. Addition of the following two products:
      i. 512 multiplied by 665
      ii. 1650 multiplied by 299
   vi. Modulus of 19111 divided by 112
   vii. Integer division of 19111 divided by 110
   viii. 9929 to the power of 120
4. Save your work (the 8 expressions in the above) in `first.py`.
5. Upload this file to your web space on the course web server and enter its URL to CourSys. The URL should be:

   http://cmpt165.csil.sfu.ca/~*userid*/first.py

Don't worry about the output you see on your web browser; we'll examine that in class on Friday.

That's it for this bonus exercise; easy credits!? Life *is* good.

Home | Simon Fraser University | Computing Science

File Edit View History Bookmarks Tools Help

Script Error

cmpt165.csil.sfu.ca/~lisat/py/first.py

csil 165web HTML Reference CSS Reference W3Cvalid WDGvalid 165.csil.sfu.ca Inter Editor coursys gsheet gdoc

Bad HTTP header

One of the HTTP headers your script outputted doesn't look right. An HTTP header must be look like `name: value`. Did you perhaps forget the blank line between the headers and contents?. You can't have anything on this line, not even a space.

This was the output:

```
I am a Python program...
...helping my author to finish a lab exercise for "CMPT 165".
14980426944
3016
309454
913028223
527265
71
1911
425266608291062138637838973134810904337149363572633512689818468381229211321580201710335636551347046877027748858815620
```

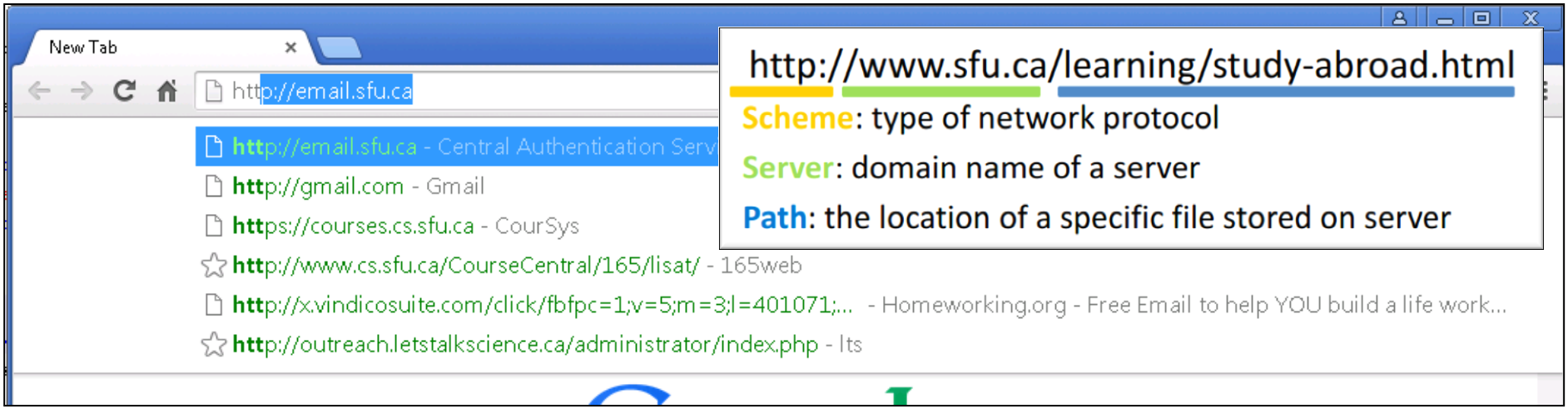
(Error code: 0)

This message was generated by CGIHelper.

FYI, error codes explained:

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#5xx_Server_Error

Fetching a resource: reviewed



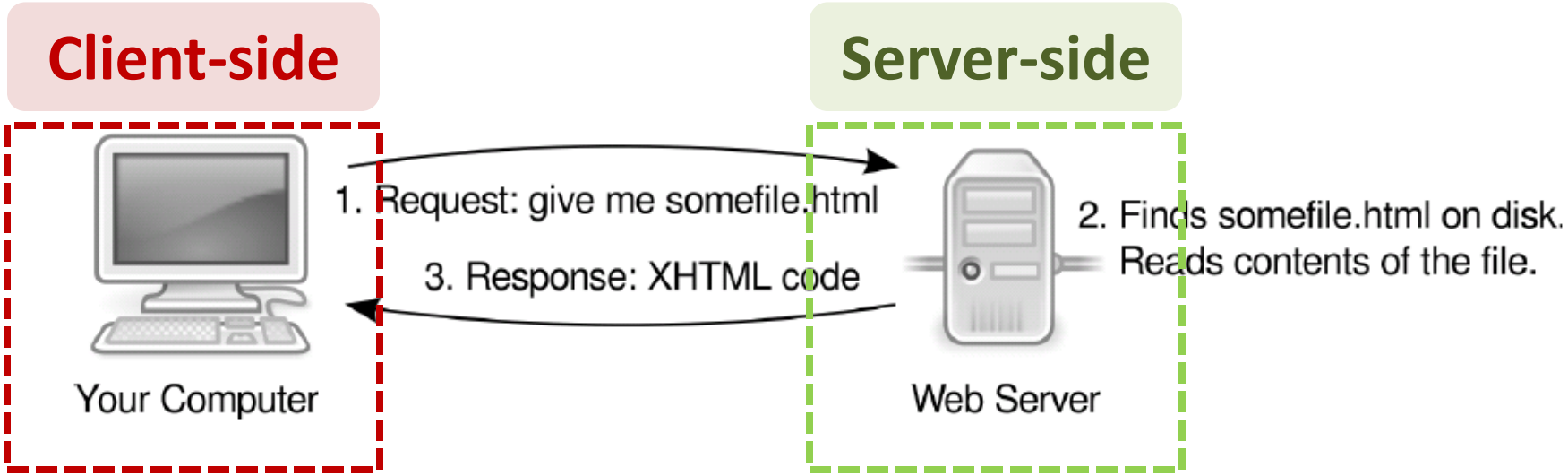


Figure 7.2: Fetching a static XHTML file

Fetching a resource: concrete example

Suppose user requests:

`http://cmpt165.csil.sfu.ca/~lisat/`

Q: Remember what the protocol is?

Ans: Procedure as follows.

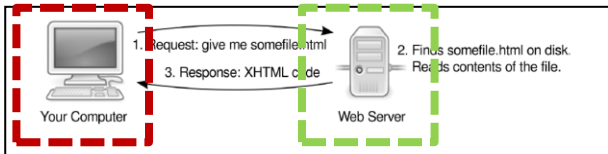


Figure 7.2: Fetching a static XHTML file

1. The user agent (users' browser) sends a request to CMPT165 server
2. The CMPT165 server sees URL has no file suffix (`html`, `htm`, `pdf`, etc.)
   i. It was **configured** to look under `public_html/lisat` (configured to look under `public_html` first and ignore ~ symbol)
   ii. It was **configured** to search for `index.html`
      ☺ If found, sends `index.html` back to user agent
      ☹ If not found, sends not found error back to user agent
3. Upon receipt, user agent either:
   ☺ Renders the markup provided in `index.html`, using the "dictionary" specified by the DOCTYPE in order to parse tags/etc. accordingly; or
   ☹ Generates "404 Not found" error on the browser winder

Fetching a resource: Part2

Suppose user requests:

`http://cmpt165.csil.sfu.ca/~lisat/py/first.py`

Q: What is the protocol then?

Ans: Procedure as follows.

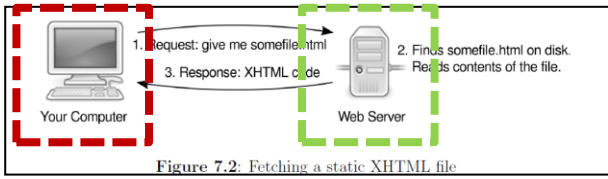


Figure 7.2: Fetching a static XHTML file

1. The user agent (users' browser) sends a request to CMPT165 server

2. The CMPT165 server sees URL has file suffix .py
   i. It was **configured** to run `first.py` in Python (installed on server)
   ii. It will perform *data processing* in `first.py`
   iii. The output of `first.py` is sent back to user agent

3. Upon receipt, user agent:
   i. Content type of received output is unknown
      →Tries to render the received data as markup
   ii. It did not find appropriate markup tags (`<head>`, etc.), so it generates "Bad HTTP header" error instead

Fetching a resource: Part2

File Edit View History Bookmarks Tools Help

Script Error

cmpt165.csil.sfu.ca/~lisat/py/first.py

csil 165web HTML Reference CSS Reference W3Cvalid WDGvalid 165.csil.sfu.ca Inter Editor coursys gsheet gdoc

Bad HTTP header

One of the HTTP headers your script outputted doesn't look right. An HTTP header must be look like `name: value`. Did you perhaps forget the blank line between the headers and contents?. You can't have anything on this line, not even a space.

This was the output:

```
I am a Python program...
...helping my author to finish a lab exercise for "CMPT 165".
14980426944
3016
309454
913028223
527265
71
1911
425266608291062138637838973134810904337149363572633512689818468381229211321580201710335636551347046877027748858815620
```

(Error code: 0)

This message was generated by CGIHelper.

MIME type

- How to resolve this?
- Specify MIME type in the output of your Python scripts
- Do so by adding this print statement:

```
print "Content-type: text/html"
print

print 'I am a Python program...'
print '...helping my author to finish a lab exercise
for "CMPT 165".'

print 5012*2988912
```

Prints blank line.
This line is required!

Static vs. dynamic resource

`*.html`
`*.htm`
`*.pdf`
`*.txt`
`*.svg`
`*.jpg`
`*.mp3`

…These are known as **static resource**: one that already exists on webserver

`*.py, *.php, *.js`

Web server…

- recognizes these as **programs** **(i.e. "web scripts")**
- use corresponding software to process these scripts and output the content generated by these scripts

Resource generated upon request known as **dynamic resource**, one created "on-the-fly"

∗ : Asterisk sign
("little star", "star" symbol)

In O/S, symbolizes *wildcard* = "anything"
`*.html` means "anything ending with .html"

Summary of key concepts/words

Program

Dynamic HTML

Developer (programmer)

Interface

GUI

Shell

Data

Input/Output (I/O)

Process

Client/server

Fetching a resource

Variables

Data Types: numeric, strings,…

Assignment (shorthand)

Operations/Operator

- Numeric: arithmetic
- Strings: Concatenation
- Overloaded symbols
- Logical (examined next week)

Functions

Statements

Misc.

- “wildcard” (*)

Questions?