Today's Agenda

- Unit 7 – Intro to programming
  - Programming essentials – Part2

- Lab exercise 8 discussed

- Pending: midterm solutions - Review #2

CMPT 165
Unit 7 – Intro to Programming - Part 4

July 22nd, 2015

SFU SIMON FRASER UNIVERSITY

Summary of key concepts & terms

Fundamentals

Developer
Interface
GUI
Shell

Program
Statements

Client/server
Fetching a resour
Dynamic HTML

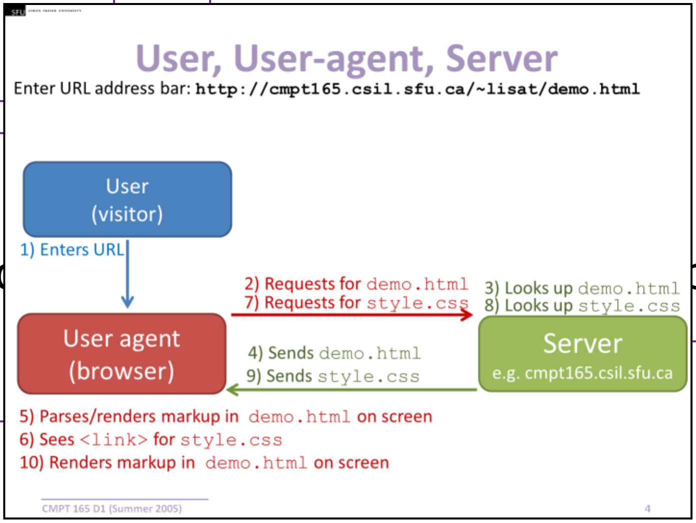
Programming essentials

Variables
Data Types
Numeric
Strings
Assignment (shorthand)

ools

Functions
Data
Input/Output (I/O)
Process

Misc.
wildcard (*)
Refining print statements


User, User-agent, Server

Enter URL address bar: `http://cmpt165.csil.sfu.ca/~lisat/demo.html`

5) Parses/renders markup in demo.html on screen
6) Sees <link> for style.css
10) Renders markup in demo.html on screen

CMPT 165 D1 (Summer 2005) 4

Review: How to print integers & strings together

Here's the syntax for this:

```
print a_number, a_string
```

```
>>> a_number=16
>>> a_string="Dear visitor, you have won $"
>>> print a_string, a_number
Dear visitor, you have won $ 16

>>> another_string=". Goodbye!"
>>> print a_string, a_number, another_string
Dear visitor, you have won $ 16 . Goodbye!
```

Summary of key concepts & terms

Fundamentals

Developer
Interface
GUI
Shell

Program
Statements

Client/server
Fetching a resource
Dynamic HTML

Programming essentials

Variables
Data Types
- Numeric
- Strings

Assignment (shorthand)

Operations/Operator
- Arithmetic
- Concatenation
- Overloaded symbols

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

Misc.
- "wildcard" (*)
- Refining print statements

function

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

Function: involves a process

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

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

- 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`**: used to print its arguments on screen

```
>>> print "Hello"
```

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

```
>>> print 'Hello'
Hello
>>> 'print'
'print'
>>> "print"
'print'
```

Review: Defining your own functions

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

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

Know the syntax!

- **Colon**
- **Indentation**

In-class Exercise: Reviewed

Given this example:

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

"isolate the variable"

Output ← Input

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

Input → Output

Q: Which is input variable to your new function?

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

```
>>> def Fahrenheit_to_Celsius(y):
        return (y-32)*5/9
```

In-class Exercise: Refined

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

- Hard to understand output
- Better output:

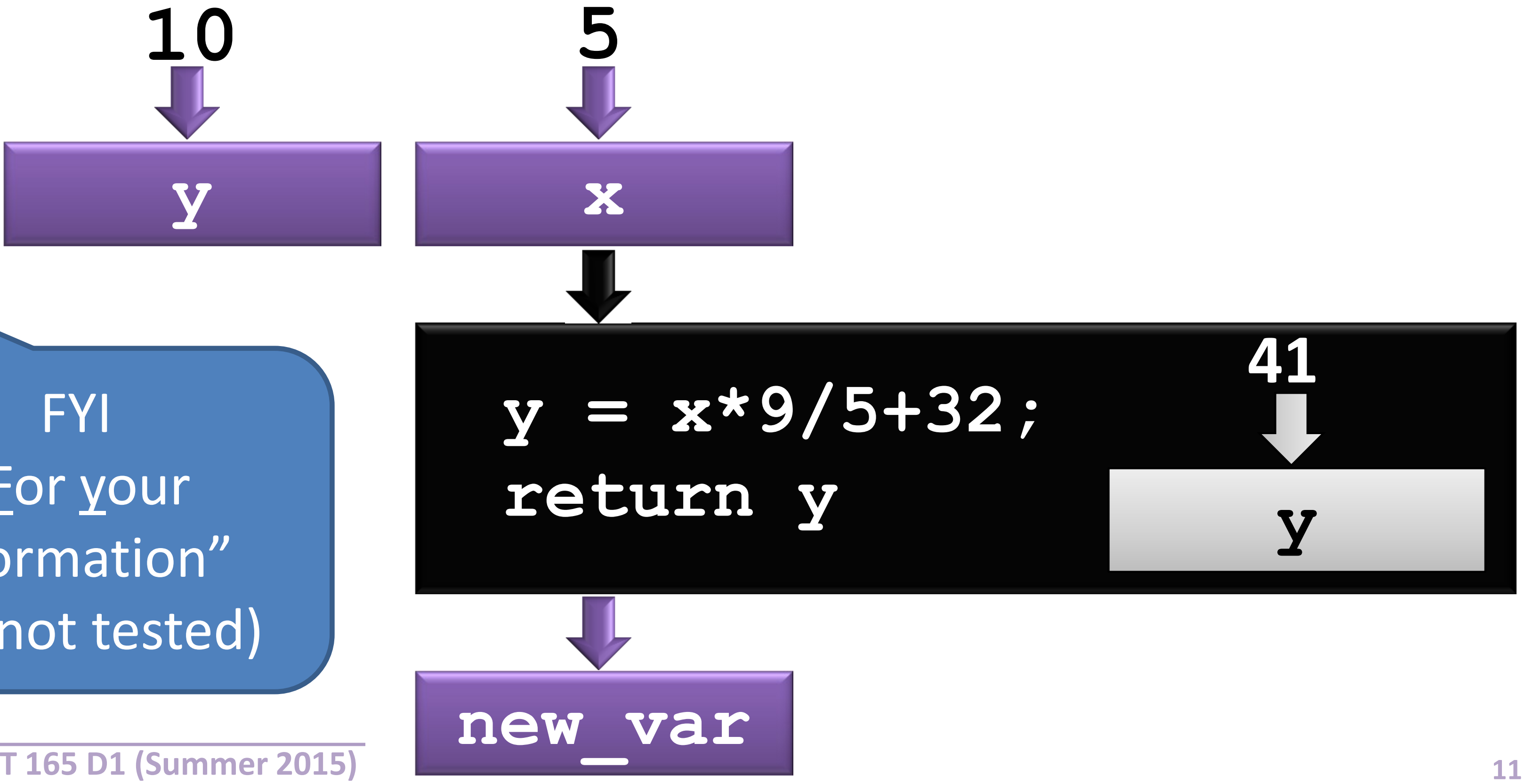
```
>>> Celsius_to_Fahrenheit(1)
1 Celsius = 25 Fahrenheit
```

- How could you code that?
- Ans: concatenating integers & strings

```
>>> def Celsius_to_Fahrenheit(x):
    print x,'Celsius =',x*9/5+32,'Fahrenheit'
```

FYI: “function as a black box”

```
>>> y=10
>>> def Celsius_to_Fahrenheit(x):
    y=x*9/5+32;
    return y
>>> x=5; new_var=Celsius_to_Fahrenheit(x)
```

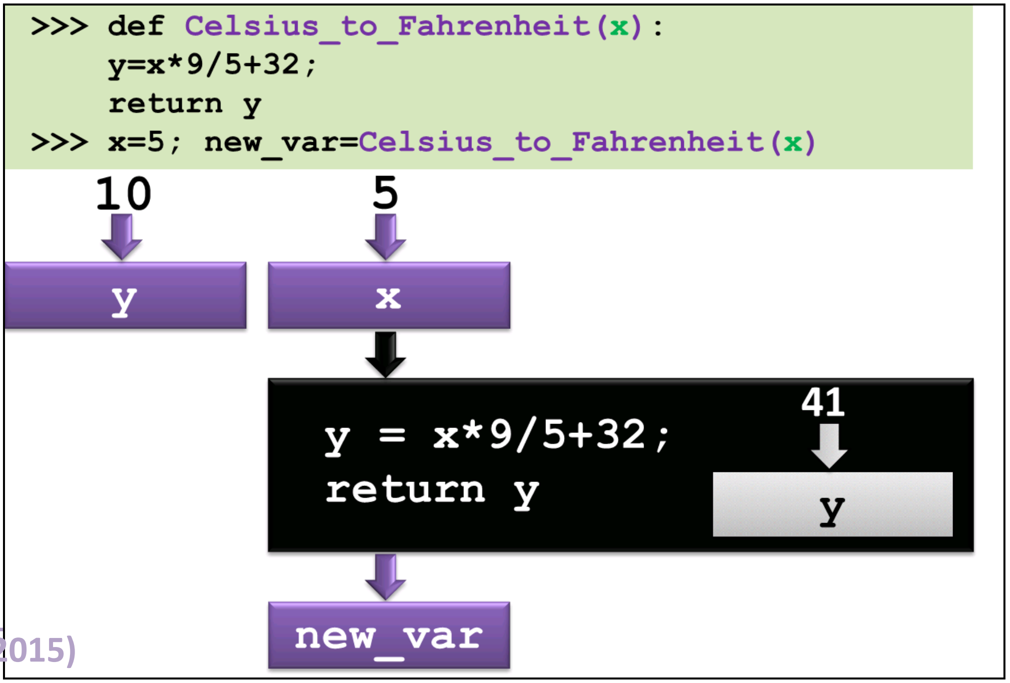


FYI
“For your information”
(i.e. not tested)

"Variable scope"

The part of a program where a variable is used is known as "scope of variables"

- e.g. scope of **y** is limited to **Celsius_to_Fahrenheit** function
- outside of this function, **y** is undefined until you instantiate (assign value to) it
- We say "**y=41** is **local** to the function"



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

Defining and using functions

Two ways.

1) In Python shell:

```
>>> y=10
>>> def Celsius_to_Fahrenheit(x):
        y=x*9/5+32;
        return y
>>> x=5; new_var=Celsius_to_Fahrenheit(x)
```

2) In IDLE Editor and run as script (e.g. `temp_convert.py`)

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

y=10;
x=5; new_var=Celsius_to_Fahrenheit(x)
```

Again:
Know the syntax!
- **Colon**
- **Indentation**

Defining and using functions

Recall: statements are executed in order saved in script (entered in Shell)

→ Functions must be defined before you can call it

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

y=10;
x=5; new_var=Celsius_to_Fahrenheit(x)
```

Defining and using functions

Would this work?

```
y=10;
def Celsius_to_Fahrenheit(x):
    y=x*9/5+32;
    return y

x=5; new_var=Celsius_to_Fahrenheit(x)
```

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

y=10;
x=5; new_var=Celsius_to_Fahrenheit(x)
```

Defining and using functions

Would this work?

```
y=10;
def Celsius_to_Fahrenheit(x):
    y=x*9/5+32;
    return y

x=5; new_var=Celsius_to_Fahrenheit(x)
```

And would this work?

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

y=10;
```

Summary of key concepts & terms

Essentials: things applicable in other programming languages

Fundamentals

Developer
Interface
GUI
Shell

Program
Statements

Client/server
Fetching a resource
Dynamic HTML

Programming essentia

Variables
Data Types
- Numeric
- Strings

Assignment (shorthand)

Operations/Operator
- Arithmetic
- Concatenation
- Overloaded symbols

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

Misc.
- wildcard (*)
- Refining print statements

Commenting in Python

Problem: Programs become large/complex

Solutions:

- Debugging: process of diagnosing problems in your code
- Add comments to explain your code
  - A good programming practice

Q: How to add comments in HTML?

```
<!-- ignored -->
```

Q: how about in CSS?

```
/* ignored */
```

```
# single line comment
var1=10;
var2=var1+3;
"""this is a multiline comment so any thing
in between is ignored """
var2*=var1;
```

Programming essentials – Part 2

(Things applicable in other programming languages)

- Commenting
- Data objects: lists
- Controlling program flow – An Intro

The `list` data object

- A **collection**: store a bunch of data types together under **one** variable name

  Examples:

```
>>> prime_nums=[2, 3, 5, 7, 11];
>>> family=['mom', 'dad', 'me', 'sister']
```

Know the syntax! Must separate each item with comma!

- You index (get/call) the 1st item as follows:

```
>>> print family[0]
mom
```

Know the syntax! Starts at 0

- Q: How to index the 3rd item in **`family`**?

  How about 5th item in **`prime_nums`?**

```
>>> family[2]
>>> prime_nums[4]
```

So, index of *n*-th item is (*n*-1)

FYI, why index starts at 0? See link: http://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html

The `list` data object

- Index groups of items, e.g. second to fourth item in **`prime_nums`**

```
>>> prime_nums=[2, 3, 5, 7, 11];
>>> prime_nums[1:3]
```

- Concatenate (join) lists:

```
>>> prime_nums=[2, 3, 5, 7, 13];
>>> prime_nums=[2, 3, 5, 7, 13] + [11, 17];
>>> print prime_nums
[1, 3, 5, 7, 11, 13, 17]
```

The `list` data object

- Has *built-in* functions
  - i.e. already in Python library*

  *software that comes along with the download when you installed IDLE on your home computer
- Syntax:

Know the syntax! Note the period

```
variable_name . method_name ( arguments )
```

```
>>> random_nums=[11, 3, 1, 5, 7];
>>> random_nums.sort()
>>> print random_nums
[1, 3, 5, 7, 11]
>>> family=['mom', 'dad', 'me', 'sister']
>>> family.sort()
>>> print family
['dad', 'me', 'mom', 'sister']
```

Programming essentials – Part 2

(Things applicable in other programming languages)

- Commenting
- Data objects: lists
- Controlling program flow – An Intro
  - Testing conditions
    - Numeric comparisons
  - If-else
  - If, else-if, else

Program flow

Idea: executes certain statements depending on conditions, e.g.

```
If (something_happens):
        Do_task1;
Else:
        Do_task2;
```

We'll see how we can get user input next class.

Concrete example (in web programming, used to govern/facilitate user-interaction), e.g.:

```
if (your_visitor_choose_to_buy_coffee):
        print '<p>', get_cost_of_coffee_to_screen('c'), '</p>'
else:
        print '<p>Thank you for visiting</p>'
```

Testing conditions

- Build complex programs by executing particular statements depending on test conditions
- Example test conditions:
  - Numerical and string comparisons:

    equal (==), less than (<), greater than (>)
- Examples:

```
>>> y=10; x=5
>>> x > y
False
>>> x > 1
True
>>> x == 5
True
>>> x == 15
False
```

```
>>> x >= 5
True
>>> y=20; x=12
>>> x > y*2
False
>>> x=1; x <= 1
True
>>> str='c'; str == 'c'
True
```

If-else

- Execute particular statements depending on condition
- Syntax:

```
if (condition_1):
    # do something
else:
    # do something else
```

- Example:

```
def compare_numbers(x,y):
    if (x > y):
        print x,'greater than',y;
    else:
        print x,'less than',y;
```

If, else-if, else

- More than 2 conditions to test?
- Syntax:

```
if (condition_1):
    # do something
elif (conditon_2):
    # do something else
else:
    # do default tasks
```

- Example:

```
def compare_numbers(x,y):
    if (x > y):
        print x,'greater than',y;
    elif (x==y):
        print x,'equal to',y;
    else:
        print x,'less than',y;
```

Practice #1

Q1) Given example conversion:

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

Write a function that takes 2 inputs:

```
    Temp_conversion(x,str)
```

And print output accordingly, like this:

```
>>> Temp_conversion(1, ‘c’)
1 Celsius = 25 Fahrenheit
>>> Temp_conversion(25, ‘f’)
25 Fahrenheit = 1 Celsius
```

Practice #2

Q2) Write a function **my_max** that returns the maximum of 3 input numbers.

```
>>> my_max(1,12,5)
12
>>> my_max(11,5,4)
11
```

```
def my_max(num1, num2, num3):

                        ?
```

Q3) Write a function **my_min** that returns the minimum of 3 input numbers.

Programming essentials – Part 2

(Things applicable in other programming languages)

- Commenting

- Data objects: lists

- Controlling program flow – An Intro
  - Testing conditions
    - Numeric comparisons
  - If-else
  - If, else-if, else

Summary of key concepts & terms

Programming essentials

Variables
- Scope of variable

Data Types
- Numeric
- Strings

Assignment (shorthand)

Operations/Operator
- Arithmetic
- Concatenation
- Overloaded symbols

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

Misc.
- "wildcard" (*)
- Refining print statements
- Commenting

Data objects: lists
- Collection of data items
- How to index items?
- Has built-in functions, e.g.

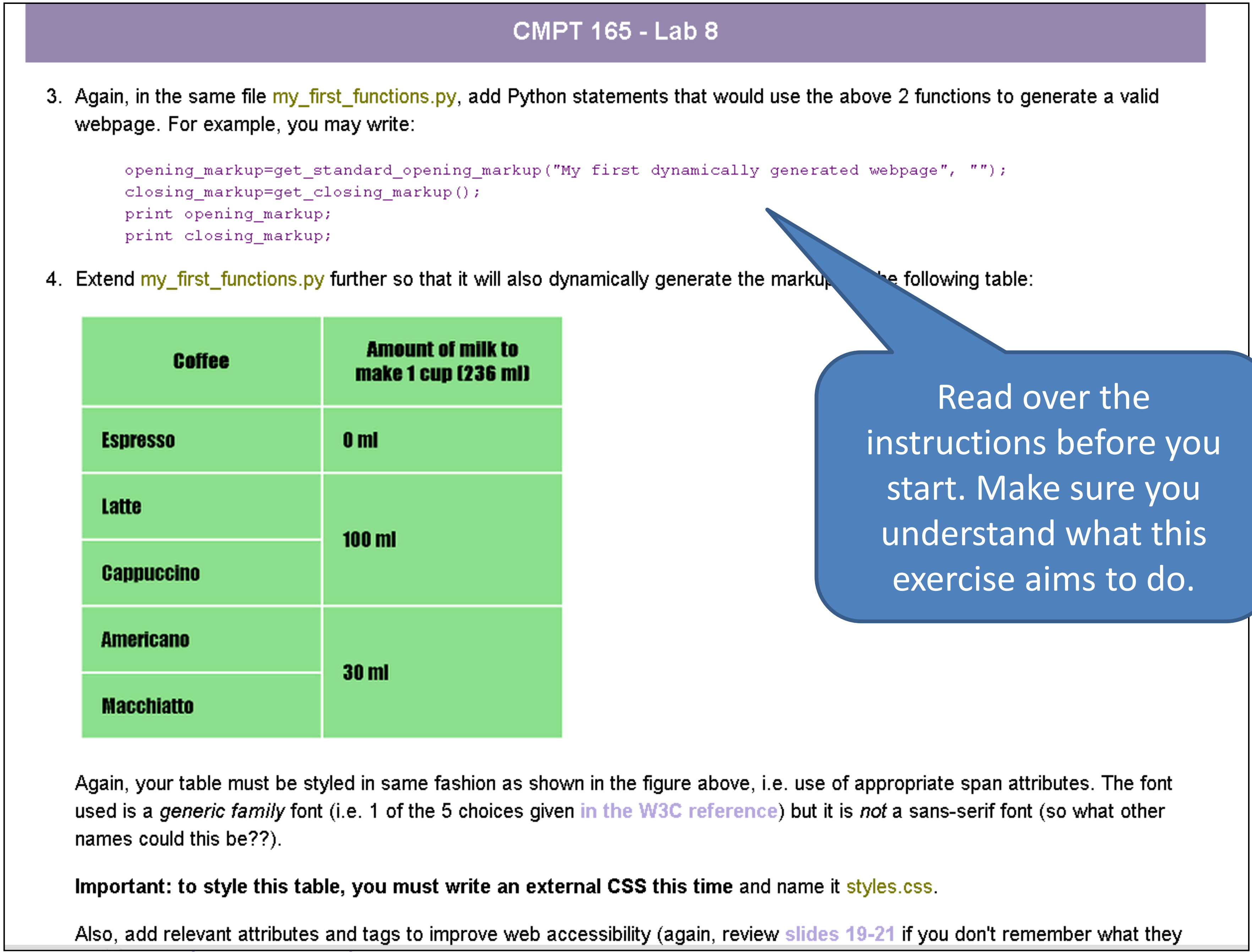
  `str.lower()`

  `numbers.sort()`

Program flow
- Test conditions
- Numeric comparisons
- String comparisons
- Control:

  `If, elif, else`

Questions?



CMPT 165 - Lab 8

3. Again, in the same file my_first_functions.py, add Python statements that would use the above 2 functions to generate a valid webpage. For example, you may write:

```
opening_markup=get_standard_opening_markup("My first dynamically generated webpage", "");
closing_markup=get_closing_markup();
print opening_markup;
print closing_markup;
```

4. Extend my_first_functions.py further so that it will also dynamically generate the marku e following table:

Coffee	Amount of milk to make 1 cup (236 ml)
Espresso	0 ml
Latte	100 ml
Cappuccino	
Americano	30 ml
Macchiatto	

Read over the instructions before you start. Make sure you understand what this exercise aims to do.

Again, your table must be styled in same fashion as shown in the figure above, i.e. use of appropriate span attributes. The font used is a *generic family* font (i.e. 1 of the 5 choices given in the W3C reference) but it is *not* a sans-serif font (so what other names could this be??).

Important: to style this table, you must write an external CSS this time and name it styles.css.

Also, add relevant attributes and tags to improve web accessibility (again, review slides 19-21 if you don't remember what they