CPS 109 - Lab 1

Thanks / Attribution

These slides and many of the Github examples were made by the head TA of CPS109 Cassandra Laffan, with edits from your TA team.

Agenda

0 - Logging In/SSH
1 - Terminology
2 - Setting Up Python
3 - IDEs/Text Editors To Use
4 - Data Types
5 - Keywords
6 - Syntax
7 - Examples

Housekeeping

There are a few things that you should know:
1 - These slides won't be posted to D2L.

2 - We have set up a GitHub Repo with some examples! You don't need to know how to use GitHub to look at them. You'll also find your weekly quizzes on here. Just click this link and go into the folders you want to see:
https://github.com/ChrisKolios/CPS109_Fall2022
3 - In this course there are students with various levels of Python experience. Please be kind to your peers in the group work and if you know the answer immediately, let your groupmates give it a go

Lab Computers

First things first, let’s get set up with the lab computers.

Username: You can find your username for the labs at my.ryerson.ca. Once you’re logged in, click the drop down list at the top right of the page (the blue icon of the person), click “Personal Account”, then click “Personal Information”. Your my.ryerson **Short ID** is your username.

Password: The default password is MMDDXXXX, where MM is your birth month, DD is your birth date, and XXXX are the **last 4 digits of your student number**.

Once you’re signed in it will prompt you to create a new password. Do so, and you will be able to log in.

Secure Shell (SSH)

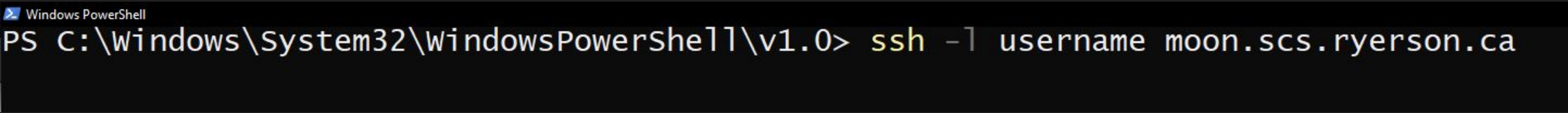
The lab suggests SSHing into a Ryerson machine.

We recommend it, too, if you want to pursue CS further (SSHing is everywhere in industry).

It’s very easy! Open your terminal (Powershell for Windows users), type in “ssh -l YourUserName ServerAddress”

Secure Shell (SSH)

In this case, to SSH into the Ryerson server:



```
PS C:\Windows\System32\WindowsPowerShell\v1.0> ssh -l username moon.scs.ryerson.ca
```

It will prompt you for your password, then voila! You’re in like hackerman.

Note that your default password will be of the form: MMDDXXXX (where MM/DD are your birth month/day, and XXXX are the last 4 digits of your student number)

Some Quick Terminology

Program - A set of instructions that a computer can interpret and execute

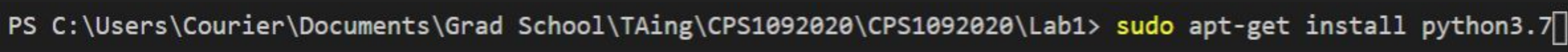
Programming/Coding - The act of writing a program

Python - The programming language we will be using for this course

Setting Up Python

Go to Python.Org and click the big download button.

If you run Linux (and somehow don’t have Python installed), you can also install it through your terminal.



```
PS C:\Users\Courier\Documents\Grad School\TAing\CPS1092020\CPS1092020\Lab1> sudo apt-get install python3.7
```

IDEs? Text Editors?

Now you’ll need some way to write Python code now that you’ve installed it. You have two choices: an IDE or a text editor.

Integrated Development Environments (IDEs)

This is where you can type code and the computer will tell you if you have typos or errors and even run the program for you.

IDLE is the default for most Python beginners and can be found at Python.org

Text Editors

You type your code into a text editor and then run it elsewhere (typically your terminal/CMD)

This option is more for students who want to learn programming while familiarizing themselves with navigating the terminal/CMD.

I suggest VSCode or Atom (both free), because they still underline typos or errors without being intrusive like an IDE.

Data Types

Integer: A whole number (e.x. 2 or -5)

Float: A “decimal” number (e.x. 4.3 or 2.22222…)

Boolean: A value of either True or False

Data Types

String: A “string” of any numbers or characters as denoted by quotes (‘’ or “”)
E.x. “Hello World” or ‘3.14’ or “:)” or ‘F’

Note: You can turn integers or floats into strings via the syntax: str(x)

Keywords

These are reserved by Python so it can follow your instructions. For example:

“and”, “or”, “not” and “return”

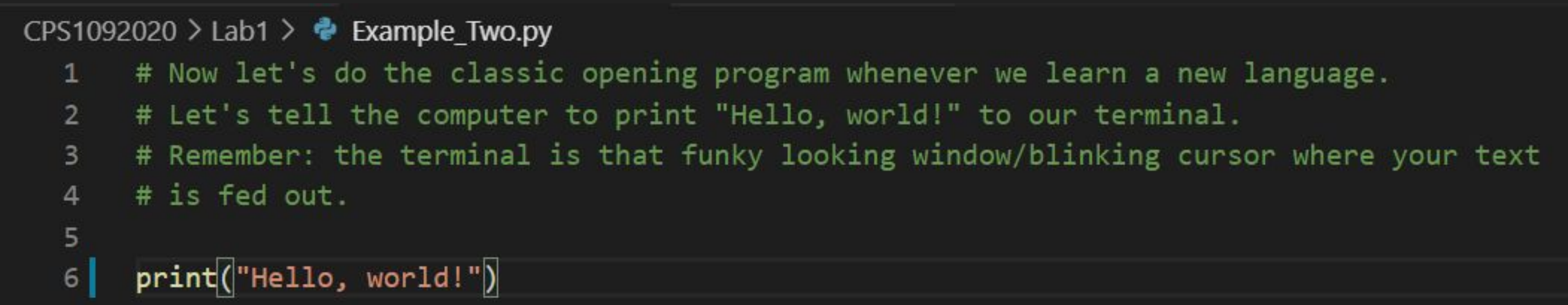
To see a list of all keywords, type help() into your python prompt and then “keywords”

Python Syntax

We have several instances of syntax examples posted on the GitHub. We encourage you to use them as references when doing your labs!

If we went over each individual syntax example one at a time, you wouldn’t have time to do your labs! So here are a few examples.

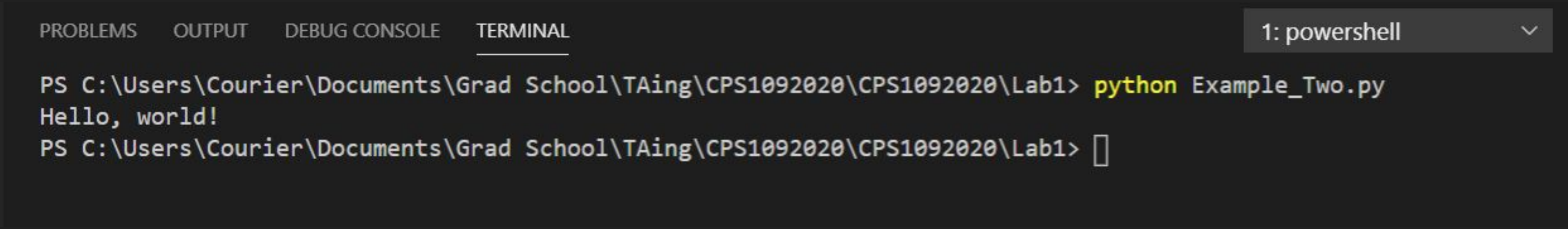
“Hello World” Example



CPS1092020 > Lab1 > Example_Two.py

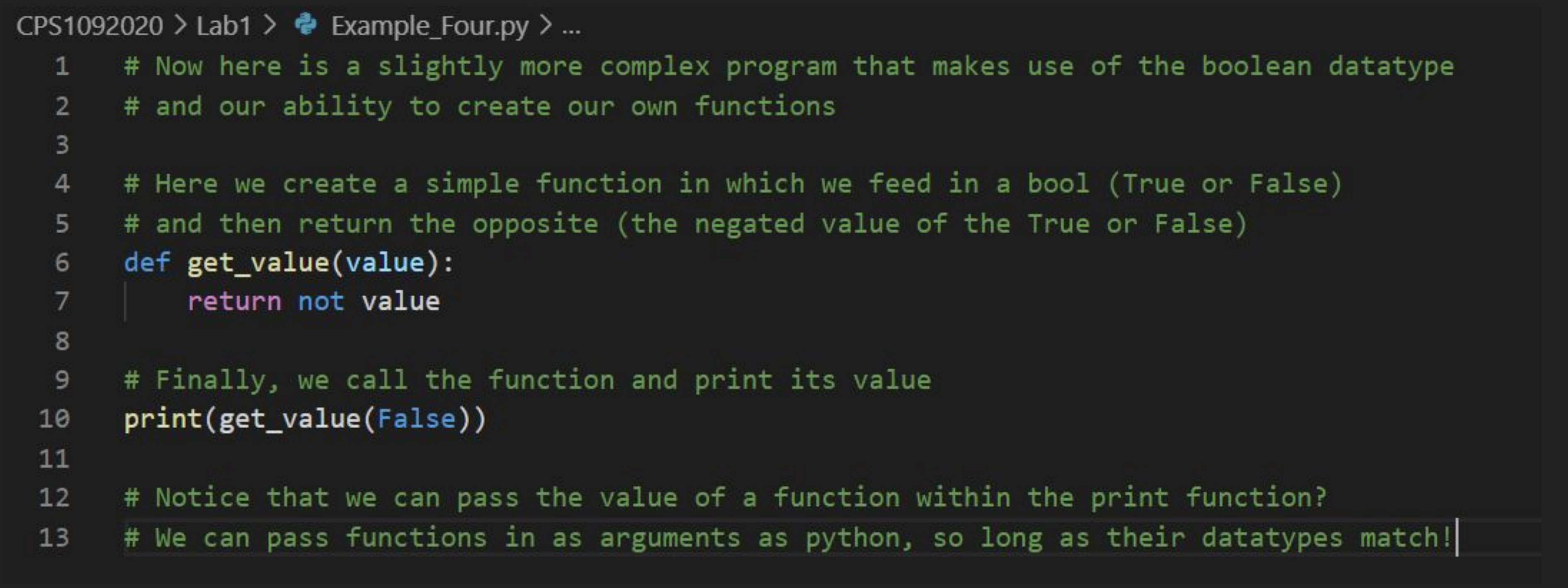
```python
# Now let's do the classic opening program whenever we learn a new language.
# Let's tell the computer to print "Hello, world!" to our terminal.
# Remember: the terminal is that funky looking window/blinking cursor where your text
# is fed out.

print("Hello, world!")
```

“Hello World” Output



```
PS C:\Users\Courier\Documents\Grad School\TAing\CPS1092020\CPS1092020\Lab1> python Example_Two.py
Hello, world!
PS C:\Users\Courier\Documents\Grad School\TAing\CPS1092020\CPS1092020\Lab1> 
```

Def Function Example


CPS1092020 > Lab1 > Example_Four.py > ...

```python
# Now here is a slightly more complex program that makes use of the boolean datatype
# and our ability to create our own functions

# Here we create a simple function in which we feed in a bool (True or False)
# and then return the opposite (the negated value of the True or False)
def get_value(value):
    return not value

# Finally, we call the function and print its value
print(get_value(False))

# Notice that we can pass the value of a function within the print function?
# We can pass functions in as arguments as python, so long as their datatypes match!
```

The Anatomy of a Function

Recall the function from the previous slide:

def is a keyword that indicates the beginning of a function

get_value is the name of this function

value is the input to this function. Any variables you want to use within the function should be here. You can have multiple variables, separated by commas
e.x. (value_1, value_2, value_3)

```
def get_value(value):
    return not value
```

don't forget the colon!

notice the indentation. Much like with if statements everything within the function should match its indentation level (or be further right, if there are internal colons)

return will exit the function, and the output will be whatever variable follows the keyword return. In this case, if you call the function with get_value(False) , your output will be: True

The Anatomy of an “if” Statement

“if” statements are an extremely useful part of most programming languages, and are vital for most programs
Consider the following example:

```
x = 3 # Declaring the variable x to be equal to 3

if (x == 3):
    print(“x is equal to 3”)

if (x <= 2):
    print(“x is less than or equal to 2”)
elif (x == 4):
    print(“x is equal to 4”)
else:
    print(“x is greater than 2 but not equal to 4”)
    if (x == 3):
        print(“and x is equal to 3!”)
print(“FIN”)
```

<u>The syntax is as follows:</u>

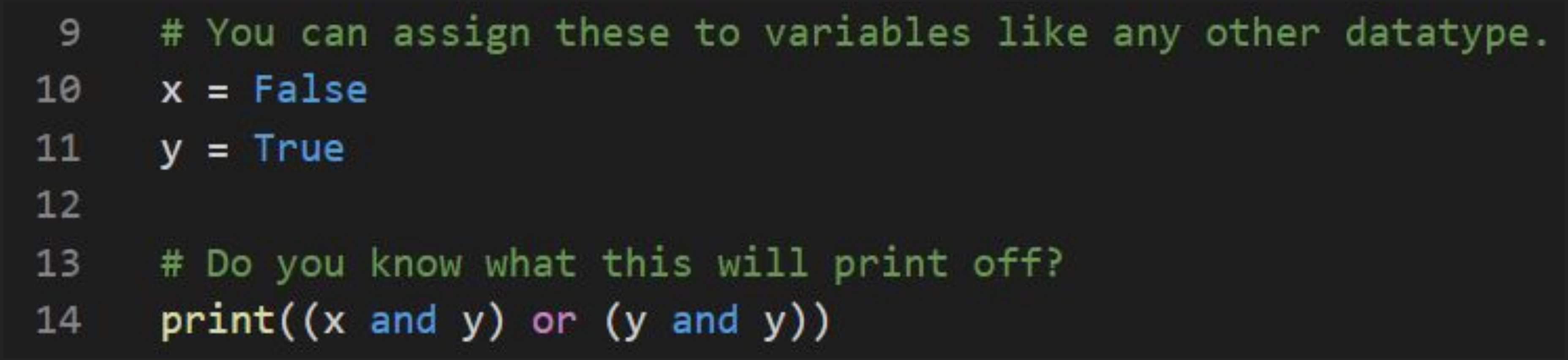
```
if/elif/else (condition):
⟷ things to do if condition is True
⟷ more things to do if condition is True
things that will happen regardless
```

Notice: indentation, the colon

<u>Example output:</u>

```
x is equal to 3
x is greater than 2 but not equal to 4
and x is equal to 3!
FIN
```

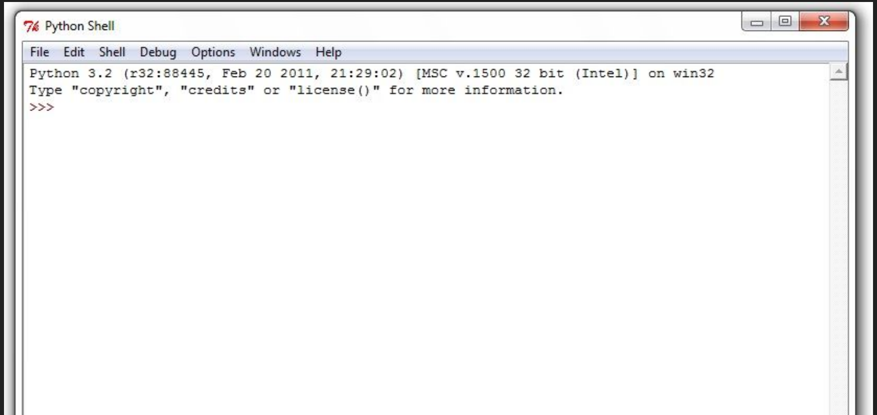
Logic Example



```python
# You can assign these to variables like any other datatype.
x = False
y = True

# Do you know what this will print off?
print((x and y) or (y and y))
```

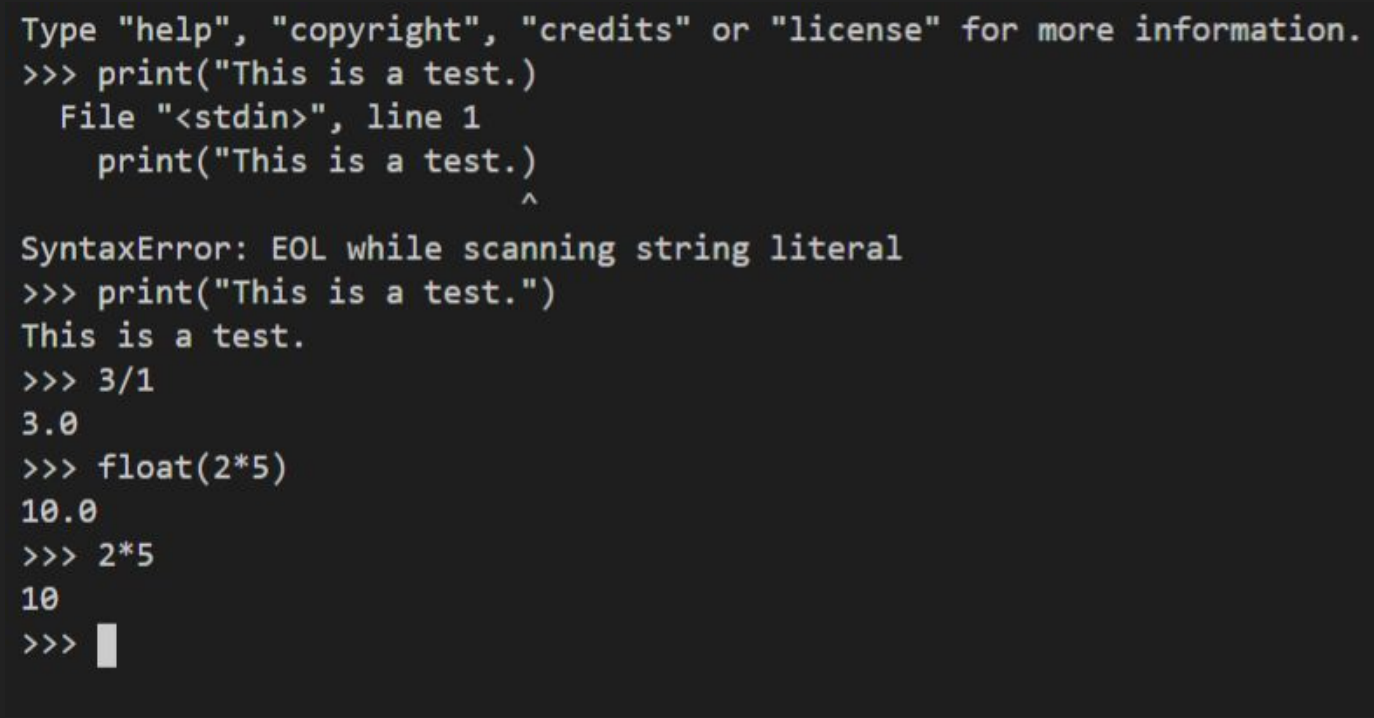
Writing in the Shell

The shell is that funky text area where you can type things in your IDE(or if you’re using a terminal, the weird prompt that comes up when you type “python”). It looks like this:



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Writing in the Shell



```
Type "help", "copyright", "credits" or "license" for more information.
>>> print("This is a test.)
  File "<stdin>", line 1
    print("This is a test.)
                          ^
SyntaxError: EOL while scanning string literal
>>> print("This is a test.")
This is a test.
>>> 3/1
3.0
>>> float(2*5)
10.0
>>> 2*5
10
>>> 
```

Contact Us

If you have any questions regarding the course content / structure / etc. (and your question is not answered by the CMF) please reach out to us at:

Chris: ckolios@ryerson.ca

Saghar: saghar.jiantavana@ryerson.ca

Course GitHub (this section):

https://github.com/ChrisKolios/CPS109_Fall2022