Objectives

- Review
- Lab 1
  - Linux practice
  - Programming practice
    - Print statements
    - Numeric operations, assignments

Reintroduce lab assistants

Student Responsibilities

- Check W&L email and course web page frequently for updates
- Actively use the interactive online text book
- Attend and participate in class and lecture
  - Be respectful to other students
- Arrive promptly to lecture/lab
  - Bring your notes and handouts
- Turn off cell phone
- Be patient, flexible, and learn from mistakes

Lab 0 Feedback

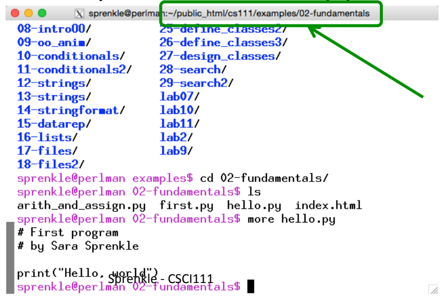
- Overall, did well
  - Lost points because didn't check work
    - E.g., broken Web page links, not including required text
  - Generally, lab grades should be high
- Interesting article links!
  - Consider reviewing for extra credit
- Sakai extra credit Easter egg
  - Great fun facts!

Lab 0 Feedback

- If there were any issues with your web page, go back and fix them first.
  - We can help!
  - Goal: Make sure you're set up for the semester

Lab 1: Linux Practice

- Review your notes, handouts from last lab
- Setting up directories
  - Make the directory, copy files
- Note: terminal tells you which directory you're in

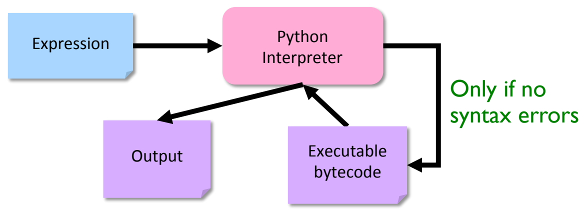


```
sprenkle@perlman:~/public_html/cs111/examples/02-fundamentals
08-intro00/            25-define_classes2/
09-oo_anim/            26-define_classes3/
10-conditionals/       27-design_classes/
11-conditionals2/      28-search/
12-strings/            29-search2/
13-strings/            lab07/
14-stringformat/       lab10/
15-datarep/            lab11/
16-lists/              lab2/
17-files/              lab9/
18-files2/
sprenkle@perlman examples$ cd 02-fundamentals/
sprenkle@perlman 02-fundamentals$ ls
arith_and_assign.py  first.py  hello.py  index.html
sprenkle@perlman 02-fundamentals$ more hello.py
# First program
# by Sara Sprenkle

print("Hello, world")
sprenkle@perlman 02-fundamentals$ 
```

Python Interpreter

1. Validates Python programming language expression(s)
   - Enforces Python syntax rules
   - Reports syntax errors ← Have a lot of these early on!
2. Executes expression(s)

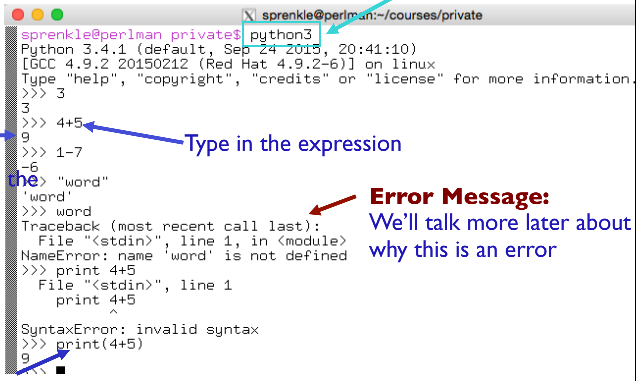


Two Modes to Execute Python Code

- **Interactive**: using the interpreter
  - Try out Python expressions
- **Batch**: execute *scripts* (i.e., files containing Python code)
  - What we'll write usually

Python Interpreter: Interactive Mode

Run by typing python3 in terminal

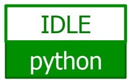


```
sprenkle@perlman:~/courses/private
sprenkle@perlman private$ python3
Python 3.4.1 (default, Sep 24 2015, 20:41:10)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 3
3
>>> 4+5
9
>>> 1-7
-6
>>> "word"
'word'
>>> word
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'word' is not defined
>>> print 4+5
  File "<stdin>", line 1
    print 4+5
          ^
SyntaxError: invalid syntax
>>> print(4+5)
9
>>> 
```

Python displays the result
Type in the expression
Error Message: We'll talk more later about why this is an error

`print`: Special *function* to display output

IDLE Development Environment



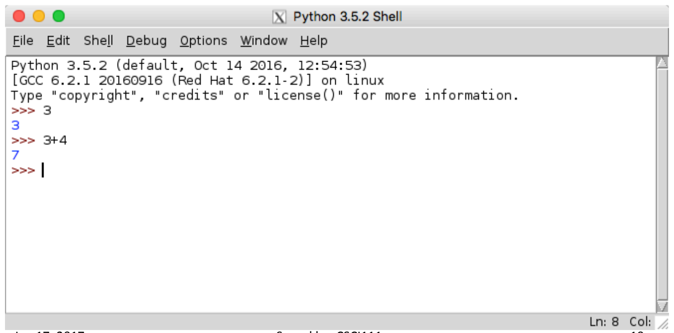
- IDLE development environment
  - Runs on top of Python interpreter
  - Command: `idle3 &`
    - `&` Runs command in "background" so you can continue to use the terminal

Since our programming language is named after Monty Python, what is the development environment named after?

- Can use IDLE to
  - Run Python in **interactive** mode
  - Write and execute scripts in **batch** mode

IDLE

- IDLE first opens up a Python shell
  - i.e., the Python interpreter in interactive mode



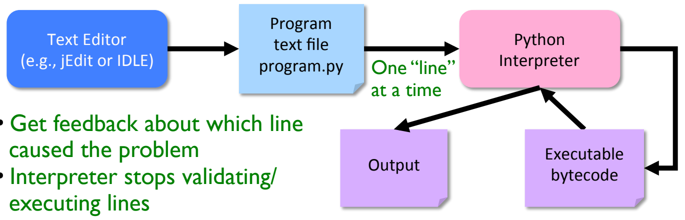
```
Python 3.5.2 (default, Oct 14 2016, 12:54:53)
[GCC 6.2.1 20160916 (Red Hat 6.2.1-2)] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 3
3
>>> 3+4
7
>>> |
```

Your Turn in Interactive Mode…

- Run `idle3` or `python3`
- Enter the following expressions and see what Python displays:
  - `3`
  - `4 * -2`
  - `-1+5`
  - `2 +`
  - `print("Hello!")`
- If you used `python3`, to quit the interpreter, use Control-D

Batch Mode

1. Programmer types a program/script into a **text editor** (jEdit or IDLE).
2. An interpreter turns each expression into bytecode and then executes each expression



- Get feedback about which line caused the problem
- Interpreter stops validating/ executing lines

Example Python Script

Text file named: `hello.py`

```
# Program that prints out "Hello, world!"
# by Sara Sprenkle, 01/17/2017

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

Print statement

- What does this program do?
  - Validate your guess by executing the program
    - Go into `/csdept/courses/cs111/lab1` directory
    - `python3 hello.py`

Example Python Script

```
# Program that prints out "Hello, world!"
# by Sara Sprenkle, 01/17/2017

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

Documentation
-- good *style*

- Only `Hello, world!` is printed out
- Python ignores everything after the "#"
  - Known as "**comments**" or, collectively, as **documentation**

Your program should *always* start with a high-level description of what the program does, your name, and the date the program was written

IDLE

- In IDLE, under the `File` menu
  - Use `New File` **or** `Open`, as appropriate, to open a window so that you can write your Python script.

Recap: Executing Python

- Interactive Mode
  - Try out expressions
  - `python3`
- Batch Mode
  - Execute Python scripts
  - `python3 <pythonscript>`
- IDLE combines these two modes into one integrated development environment

Review

- How do we display output?
- What are the data types available in Python?
- How should we name variables?
- How do we assign values to variables?

Recap: Programming Fundamentals

- Most important data types (for us, for now): `int, float, str, bool`
  - Use these types to represent various information
- Variables have identifiers, (implicit) types
  - Should have "good" names
  - Names: start with lowercase letter; can have numbers, underscores
- Assignments
  - `x = y` means "x set to value y" or "x is assigned value of y"
  - Only variable on LHS of statement changes

Review: Assignment statements

- Assignment statements are NOT math equations!

```
count = count + 1
```

- These are commands!

```
x = 2
y = x
x = x + 3
```

What is the value of y?

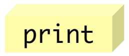
Numeric Arithmetic Operations

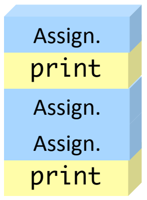
Symbol	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder ("mod")
**	Exponentiation (power)

Remember PEMDAS

Programming Building Blocks

- Each type of statement is a building block
  - Initialization/Assignment
    - So far: Arithmetic
  - Print
- We can combine them to create more complex programs
  - Solutions to problems

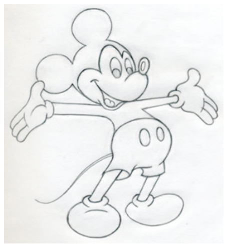




Formalizing Process of Developing Computational Solutions

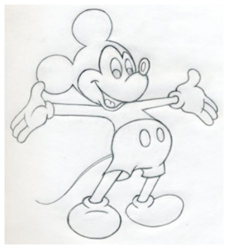
1. Create a sketch of how to solve the problem (the algorithm)

Use comments to describe the steps



Formalizing Process of Developing Computational Solutions

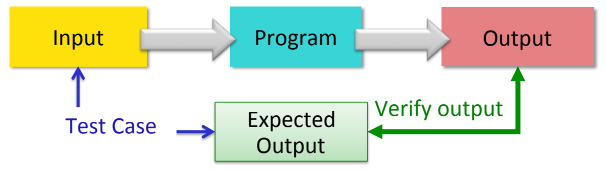
1. Create a sketch of how to solve the problem (the algorithm)
2. Fill in the details in Python



Errors

- Sometimes the program doesn't work
- Types of programming errors:
  - Syntax error
    - Interpreter shows where the problem is
  - Logic/semantic error
    - answer = 2+3
    - No, answer should be *2*3*
  - Exceptions/Runtime errors
    - answer = 2/0
    - Undefined variable name

Testing Process

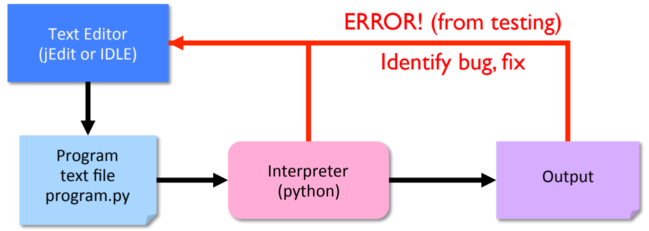


- Test case: **input** used to test the program, **expected output** given that input
- Verify if output is what you expected

If output is not what you expect…

Debugging

- After identifying errors during *testing*
- Identify the problems in your code
  - Edit the program to fix the problem
  - Re-execute/test until all test cases pass
- The error is called a "bug" or a "fault"
- Diagnosing and fixing error is called ***debugging***



Lab 1: Programming Practice

- After the warm up problems
- Name program files **lab1.n.py**, where *n* is the problem you're working on
- After completed, demonstrate that your program works
  1. Close IDLE/Python interpreter, rerun program
     - Get rid of the output from when you were developing/ debugging ("scratch work")
  2. Save output for each program in file named `lab1.n.out` where *n* is the problem you're working on

Lab 1 Expectations

- Comments in programs
  - High-level comments, author
  - Notes for your algorithms, implementation
- Nice, readable, understandable output
  - User running your program needs to understand what the program is saying
- Honor System
  - Pledge the Honor Code on printed sheets

Lab 1 Submission

- Electronic as well as printed
  - I can execute your program, help find mistakes
  - Copy your lab directory into your `turnin` directory
- Instructions are in the lab