• Labs have started
• Homework 1 is posted (Due Friday)
Administrivia

• i>clicker
  – Starting testing today
  – Need to register device **ON COURSE COMPASS PAGE**
A set of instructions a computer executes to achieve a goal is called…

a) a process.
b) a program.
c) a procedure.
d) a pronoun.
A grouping of 8 bits is called…
a) a nibble.
b) a chomp.
c) a byte.
d) a gobble.
Python is…
a) a high-level language.
b) a low-level language.
Python is…
a) a compiled language.
b) an interpreted language.
BASIC PYTHON SYNTAX AND SEMANTICS
Computer
Literals

• Describe data that *doesn’t change*
• **ANALOGY**: Literals are *nouns* in Python
• Represent a fixed *value* (e.g. 3 or 5,136,833,998)
Computer

Processor

3
Operators

• Describe how to *manipulate* data
• **ANALOGY**: Operators are the *verbs* of Python
• Common mathematical operators (e.g. +, -, *, /) are operators
• There are *many* more operators
Computer

+  

Processor
Expressions

• Combining constants and operators, we can build *expressions*

• **ANALOGY**: Expressions are *sentence fragments* in Python

• Expressions are *evaluated* to produce a new value (e.g. $3 \times 5$ or $23 - 100$)

• Expressions can be very complicated (e.g. $3 + 8 \times 5 + 4 - 7 / 100$)
Computer

Processor

15
Order of Operations

• $1+1\times2$
  a) 4
  b) 3
  c) None of the above

• Like math, Python has order of operations
• Not always intuitive
• When in doubt, use parentheses!
Evaluate this expression:
\[23 + \frac{6}{2} - 4\]

a) 22
b) 18
c) -9
d) None of the above
Other operators

• Modulo
  – Symbol: %
  – Description: remainder after division
  – Example: 9 % 2

• Exponentiation
  – Symbol: **
  – Description: base to the exponent power
  – Example: 3 ** 2
Evaluate this expression:

\[(28 \% 5)^3\]

a) 8  
b) 27  
c) 64  
d) None of the above
Bitwise operators

- YOU DON’T NEED TO KNOW THESE
- Operate on *binary* representation
- Bitwise or |
- Bitwise xor ^
- Bitwise and &
- Shift left <<
- Shift right >>
Evaluate this expression:

1^2

a) 0
b) 1

c) 2
d) 3

THIS IS WRONG!!!
Computer

Processor
Problem

- Computer is in the same *state* as when we started
- Programs are *complex*
- We computer to *remember* the results
- We need to *store* the resulting value
Computer

Processor
Computer

Memory

Processor

15
Problem

• How do we know where data “lives”?  
• In low-level languages, data has an “address” represented in binary

ADD DATA AT 1010110111010100  
TO DATA AT 1101010001001001  
STORE RESULT 0000110101001110  
YUCK!!!!
Solution

• Give the memory locations a “name”
Variables

- A name for a memory location used to store data
- **ANALOGY**: Variables are *nouns* in Python.
- Variables store a *value*
- The value stored in a variable *can change over time*
- A variable is a place holder
Assignment

• Stores a value in a variable (memory)
• Uses the = symbol
  – Variable on the left
  – Expression on the right
• Example: x=3 stores the value 3 in variable x
• *Defines* (names) the variable if we have not already used it
\[ x = 15 + 7 \times 9 \]

What value is stored in variable \( x \)?

a) 3
b) 31
c) 55
d) 78
x = 15 + 7 * 9
x = 3

What value is stored in variable x?

a) 3
b) 31
c) 55
d) 78
Statement

• A statement changes the state of the computer
• **ANALOGY**: Statements are *sentences* in Python.
• An *assignment* is a statement
• Our programs will be a series of statements
Script

• A file containing a series of Python statements
• Stored in text (no magic, *just* text)
• Each instruction is executed top to bottom
  – Starting from the first line
• Together, the statements form a program
Example Program

\[
x = 10 \\
y = x^{**2} \\
y = y + y
\]