Assignment

- First program due tomorrow night.
- The tricky part: how can you avoid using the digits 1,2,3 and just use the digit 4? But your program is supposed to print 1,2 and 3?
- Using just the print statement.
- Why? To show that there are lots of ways to write a program that does the same thing.

Submission

- SmartSite should now be configured properly to accept programs.
- Submit as attachment.
- Once you click submit, you cannot re-submit.

The print statement

- Prints things.
- Can print text given in quotes:
  ```python
  print "Take me to your leader."
  ```
  Important vocabulary word
down arrow

  "Take me to your leader." is a string. Strings are green in IDLE (on Windows).

Strings in English

- Dick said, "See what Mike and I have. This is Puff with me. That is Spot with Mike."

Programming jargon

- This is a statement.
- print is a Python command.
- "Take me to your leader" is a string
- "9/7" is also a string, because it's in quotes
- "9/7" is treated as a piece of text, Python does not do arithmetic.
The print statement

- You can also print numbers, and use arithmetic.
  
  ```python
  print 4+4
  ```
- **4+4** is an arithmetic expression.
- It's value is 8.

Running IDLE

- From START menu, pick “all programs”, then Python
- Pick the “IDLE” option

IDLE

- IDLE is an interpreter.
- Responds to input line-by-line

Floating point numbers

- 7.0, 2.0, 0.0006, 7.34 – **floating point numbers**
- 7.0/2.0 is **floating point division**. It produces the value 3.5.
- If either number is floating point, so is the answer – so 7.0/3 produces the value 3.5

Remainder

- 0, 1, 2, 3… and -1, -2, -3…. are **integers**
- 7/3 is **integer division**
- 7%2 = 1
- % gives the remainder when 7 is divided by 2
- (7/2)*2 + (7%2) = 7

Data types

- Integers and floating point numbers are two different **data types** in Python
- Can do different things with them.
- 3.0 is different from 3
- 3/2 is different from 3.0/2.0
- "3.0" is different from 3.0
Floating point is not exact

>>> 8.0/3.0
2.6666666666666665

- This is weird...why?
- Computer numbers have a fixed number of decimal places
- Exact results with floating point numbers would have an infinite number of decimal places:
  Example: 8.0/3.0 has the value 2.666666......

Variables

>>> x = 2.0
>>> x = x+3.0
>>> x
5.0

- x is a **variable**
- "x = 2.0" is called an **assignment** statement
- Variable on left-hand side gets value on right-hand side.
- Pronounce this "x gets 2.0" or "x becomes 2.0"

Variables

- Thinking of a variable as a box.

- Assignment statement puts a data value into variable x, it is not itself a data value or an expression.

- The name x is the label on the box.

Errors

- Lots of things you do will cause errors
- Something Python doesn’t understand

- Error because variable is supposed to be on the left.

2.0

- The floating point number 2.0 is in the box.
Errors

\[ y = y + 3 \]

- Error because you ask it to give the value \( y + 3 \) to \( y \), but it doesn't know what \( y \) is.
- Variables don’t stand for “any old number” like they do in algebra; a variable must always have a specific value.

Making a program

- Do something more complicated
- Remember and repeat a bunch of commands

A program

- A program is a list of statements in a file
- Python executes the statements one by one

Your program

- Uses print
- Uses some arithmetic?