Variables

- When creating a variable in Python, no type is provided
  - `x = 20`
  - `s = 'You think youve lost your love'
  - `pi = 3.14159`

- So what happens if I do this (after the statements above)?
  - `x = pi`
  - Answer: `x` has the value `3.14159`
Types

- Python has types – what determines the type of an object?
- Duck typing
  - if it walks like a duck and quacks like a duck...

So on previous slide, \( x \) was an int until we reassigned it. Then it became a float.

- `type(x)` reports the type of variable \( x \)

Built-in types

- boolean
  - values `True` and `False` (note capitalization)
- Numeric types: int, float, long, complex
- Sequence types: str, list, tuple
- set
- dict
Truth values

• Any object can be used in a Boolean expression
• The following evaluate to False
  • None
  • False
  • 0
  • empty sequences: '', [], ()
  • empty dictionary: {}

Type Casting

• Syntax: new_type(object)

\[
\begin{align*}
  x &= 3.14159 \\
  y &= 20 \\
  s &= '20' \\
  \text{int}(x) &\quad \rightarrow \quad 3 \\
  \text{float}(y) &\quad \rightarrow \quad 20.0 \\
  \text{str}(x) &\quad \rightarrow \quad '3.14159' \\
\end{align*}
\]
Type Casting continued

- $x = 3.14159$
- $y = 20$
- $s = '20'$
- $t = '111'$

- `int(s)` → 20
- `int(y)` → error
- `str(x)` → '3.14159'
- `int(t, 2)` → 7
- `int(t, 8)` → 73

Operator Precedence

- No surprises
  - parentheses
  - exponentiation
  - multiplication/division
  - addition/subtraction

- "I don’t work very hard to remember rules for other operators. If I can’t tell by looking at the expression, I use parentheses to make it obvious."
String Operators

• We will see string methods later. Here are a couple of useful operators.

• + string concatenation
  • s1 = ‘This is ‘
  • s2 = ‘a test.’
  • print s1 + s2

• * string repeat
  • s = ‘spam ‘
  • print s * 4 + ‘eggs, bacon, and spam’