Introduction to Computer Programming

Lecture 4
Numbers and Expressions
Topics

• Review basic math operators
• More math operators (\,\, %, **) 
• Complex expressions and order of elaboration 
• Math formulas and Python expressions 
• More on printing (long lines, format)
Review: Good Programming Style

- Meaningful comments throughout program
- Meaningful variable names
- Each variable name should have a single purpose and type
- Avoid numeric constants in program body – assign them to variables at top of program (e.g. Pi, rates, …)
- Use spacing well (blank lines between program sections)
- No more than 80 characters per line
- Clear logic
Tip: Breaking Long Statements into Multiple Lines

- For readability, a line should not be longer than 80 characters
  - Long statements cannot be viewed on screen without scrolling and cannot be printed without cutting off

- Multiline continuation character (\): Allows to break a statement into multiple lines

```python
print('my first name is', \first_name)

my first name is Edith

print('abcdefghi\jklmnop')

abcdefg hijklmnop
```
How to Test A Program

• Always run your program – never assume it just works.

• Test your program on different inputs
  – Try easy inputs (e.g. 90, 80, 100 for average)
  – Try hard inputs
    • Make sure you know what the answer should be
  – Test all possible scenarios and corner cases
    • e.g., for dates, make sure works for 1900’s and 2000’s

• Test on illegal inputs
  – See if you can crash the program (programs shouldn’t crash!)
  – Need to program defensively
# Review: Arithmetic Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>Adds two numbers</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>Subtracts one number from another</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>Multiplies one number by another</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>Divides one number by another and gives the result as a floating-point number</td>
</tr>
</tbody>
</table>
Review: Arithmetic Expression

- **Operand**: Integer, Float
- **Operator**: +, -, *, /
Result type rule for Arithmetic Expressions

The result of Integer arithmetic is always an Integer, except when operation is division.

- \[ \text{Integer} \ operator \ \text{Integer} \rightarrow \text{Integer} \]
- \[ \text{Integer} / \ \text{Integer} \rightarrow \text{Float} \]

Otherwise, the result is floating point:

- \[ \text{Integer} \ operator \ \text{Float} \rightarrow \text{Float} \]
- \[ \text{Float} \ operator \ \text{Float} \rightarrow \text{Float} \]
- \[ \text{Float} \ operator \ \text{Integer} \rightarrow \text{Float} \]
Quiz: Is grade an **Integer** or a **Floating Point** number?

\[
\text{grade} = 15 + 27 \\
\text{grade} = 15 + 1.2 \\
\text{grade} = 1.0 + 2.0 \\
\text{grade} = 100 \times .66 \\
\text{grade} = 100 - 2 \\
\text{grade} = 100 - 2.0 \\
\text{grade} = 100 / 3
\]
Quiz: Is grade an **Integer** or a **Floating Point** number?

\[
\text{grade} = 15 + 27 \quad \# \text{ Integer}
\]

\[
\text{grade} = 15 + 1.2
\]

\[
\text{grade} = 1.0 + 2.0
\]

\[
\text{grade} = 100 \times .66
\]

\[
\text{grade} = 100 - 2
\]

\[
\text{grade} = 100 - 2.0
\]

\[
\text{grade} = 100 / 3
\]
Quiz: Is grade an **Integer** or a **Floating Point** number?

grade = 15 + 27  # Integer

grade = 15 + 1.2  # Floating Point

grade = 1.0 + 2.0

grade = 100 * .66

grade = 100 – 2

grade = 100 – 2.0

grade = 100 / 3
Quiz: Is grade an **Integer** or a **Floating Point** number?

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>grade = 15 + 27</code></td>
<td>Integer</td>
</tr>
<tr>
<td><code>grade = 15 + 1.2</code></td>
<td>Floating Point</td>
</tr>
<tr>
<td><code>grade = 1.0 + 2.0</code></td>
<td></td>
</tr>
<tr>
<td><code>grade = 100 * .66</code></td>
<td></td>
</tr>
<tr>
<td><code>grade = 100 - 2</code></td>
<td></td>
</tr>
<tr>
<td><code>grade = 100 - 2.0</code></td>
<td></td>
</tr>
<tr>
<td><code>grade = 100 / 3</code></td>
<td></td>
</tr>
</tbody>
</table>
Quiz: Is grade an **Integer** or a **Floating Point** number?

- grade = 15 + 27  
  # Integer
- grade = 15 + 1.2  
  # Floating Point
- grade = 1.0 + 2.0  
  # Floating Point
- grade = 100 * .66
- grade = 100 – 2
- grade = 100 – 2.0
- grade = 100 / 3
Quiz: Is grade an **Integer** or a **Floating Point** number?

- grade = 15 + 27  # Integer
- grade = 15 + 1.2  # Floating Point
- grade = 1.0 + 2.0  # Floating Point
- grade = 100 * .66  # Floating Point
- grade = 100 – 2  # Integer
- grade = 100 – 2.0  # Floating Point
- grade = 100 / 3  # Floating Point
Quiz: Is grade an **Integer** or a **Floating Point** number?

\[
\text{grade} = 15 + 27 \quad \# \text{ Integer}
\]
\[
\text{grade} = 15 + 1.2 \quad \# \text{ Floating Point}
\]
\[
\text{grade} = 1.0 + 2.0 \quad \# \text{ Floating Point}
\]
\[
\text{grade} = 100 \times .66 \quad \# \text{ Floating Point}
\]
\[
\text{grade} = 100 - 2 \quad \# \text{ Integer}
\]
\[
\text{grade} = 100 - 2.0
\]
\[
\text{grade} = 100 / 3
\]
Quiz: Is grade an **Integer** or a **Floating Point** number?

- \( \text{grade} = 15 + 27 \)  # Integer
- \( \text{grade} = 15 + 1.2 \)  # Floating Point
- \( \text{grade} = 1.0 + 2.0 \)  # Floating Point
- \( \text{grade} = 100 \times 0.66 \)  # Floating Point
- \( \text{grade} = 100 - 2 \)  # Integer
- \( \text{grade} = 100 - 2.0 \)  # Floating Point
- \( \text{grade} = 100 \div 3 \)
Quiz: Is grade an **Integer** or a **Floating Point** number?

\[
\begin{align*}
\text{grade} &= 15 + 27 & \# \text{ Integer} \\
\text{grade} &= 15 + 1.2 & \# \text{ Floating Point} \\
\text{grade} &= 1.0 + 2.0 & \# \text{ Floating Point} \\
\text{grade} &= 100 \times .66 & \# \text{ Floating Point} \\
\text{grade} &= 100 - 2 & \# \text{ Integer} \\
\text{grade} &= 100 - 2.0 & \# \text{ Floating Point} \\
\text{grade} &= 100 / 3 & \# \text{ Floating Point}
\end{align*}
\]
## More Arithmetic Operations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>Division</td>
<td>Divides one number by another and gives the result as a floating-point number</td>
</tr>
<tr>
<td>//</td>
<td>Integer division</td>
<td>Divides one number by another and gives the result as an integer</td>
</tr>
<tr>
<td>%</td>
<td>Modulo (mod)</td>
<td>Divides one number by another and gives the remainder</td>
</tr>
<tr>
<td>**</td>
<td>Exponent</td>
<td>Raises a number to a power</td>
</tr>
</tbody>
</table>
Two types of division

• print (5/2)    #  2.5
• print (5//2)   #  2
• print (-5/2)   # -2.5
• print (-5//2)  # -3

Truncation rounds towards zero for positive numbers, and away from zero for negative numbers.
Remainder % (mod) operator

X % N results in a number between 0 and N-1

• $X \% 2$  # 0 or 1

• $X \% 12$  # 0 thru 11

• $X \% 60$  # 0 thru 59
// and % work well together

# Are we there yet?
# Calculate how much time to go assuming you are
# going 60 mph (a mile a minute)

miles = int(input("How many miles to go? "))
minutes = miles
hours = minutes // 60
remaining = minutes % 60
print(hours, "hours and", remaining, "minutes more")
Mod % is surprisingly useful!

• Is a number even or odd?
• Arithmetic on anything that is periodic
  – Days of the week (M T W Th F Sa Su M T W Th, F..)
  – Hours of the day, clock time, months of the year
  – Musical scale (do, re, mi, fa, sol, la, ti, do, re, mi,..)
• Taking turns in games
• Running tasks on a cycle
• Cryptography
  ......
Exponent Operation

• An exponent operator raises a number to a power

\[ x^{10} \text{ becomes } x^{**}10 \]

EXAMPLE

```python
square_side = float(input('Length of square side: '))
square_area = square_side**2
print('Area: ', square_area)
```
Expressions with Multiple Operators

What is the answer?

\[ 10 \div 2 \times 5 \]

Advice: When in doubt, use parentheses!

\[ (10 \div 2) \times 5 \]
\[ 10 \div (2 \times 5) \]

For the maven (and exams), know the rules!
Rules for Order of Elaboration
(also called, Operator Precedence Rules)

Higher precedence operators executed first

Highest: Operations enclosed in parentheses
Next Highest: Exponentiation (**)
Next: Multiplication (*), division (/ and //), and remainder (%)
Lowest: Addition (+) and subtraction (-)

– Same precedence operators execute from left to right
– USE PARENTHESES TO OVERRIDE PRECEDENCE ORDER
Example

\[
90 + 80 + 100 / 3
\]

\[
90 + 80 + 33.333
\]

\[
170 + 33.333
\]

203.333
Example

\[
\frac{4}{2} * 2^2
\]

\[
\frac{4}{2} * 4
\]

\[
2 * 4
\]

8
Example: Order of Operation

2 + 4 * 47 / 2 ** 3 - 5 * (1 – 3 + 4)
Example: Order of Operation

\[ 2 + 4 \times 47 / 2 \times 3 - 5 \times (1 - 3 + 4) \]
\[ 2 + 4 \times 47 / 2 \times 3 - 5 \times (-2 + 4) \]
Example: Order of Operation

\[ 2 + 4 \times 47 / 2 \times 3 - 5 \times (1 - 3 + 4) \]

\[ 2 + 4 \times 47 / 2 \times 3 - 5 \times (-2 + 4) \]

\[ 2 + 4 \times 47 / 2 \times 3 - 5 \times 2 \]
Example: Order of Operation

\[2 + 4 \times 47 / 2 \quad ** \quad 3 - 5 \quad * (1 - 3 + 4)\]

\[2 + 4 \times 47 / 2 \quad ** \quad 3 - 5 \quad * (-2 + 4)\]

\[2 + 4 \times 47 / 2 \quad ** \quad 3 - 5 \quad * 2\]

\[2 + 4 \times 47 / 8 - 5 \quad * 2\]
Example: Order of Operation

\[2 + 4 \times 47 / 2 \times 3 - 5 \times (1 - 3 + 4)\]
\[2 + 4 \times 47 / 2 \times 3 - 5 \times (-2 + 4)\]
\[2 + 4 \times 47 / 2 \times 3 - 5 \times 2\]
\[2 + 4 \times 47 / 8 - 5 \times 2\]
\[2 + 188 / 8 - 5 \times 2\]
Example: Order of Operation

\[ 2 + 4 \times 47 / 2 \quad \times 3 \quad - 5 \quad \times (1 - 3 + 4) \]
\[ 2 + 4 \times 47 / 2 \quad \times 3 \quad - 5 \quad \times (-2 + 4) \]
\[ 2 + 4 \times 47 / 2 \quad \times 3 \quad - 5 \quad \times 2 \]
\[ 2 + 4 \times 47 / 8 \quad - 5 \quad \times 2 \]
\[ 2 + 188 / 8 - 5 \times 2 \]
\[ 2 + 23.5 - 5 \times 2 \]
Example: Order of Operation

2 + 4 * 47 / 2 ** 3 - 5 * (1 – 3 + 4)
2 + 4 * 47 / 2 ** 3 - 5 * (-2 + 4)
2 + 4 * 47 / 2 ** 3 - 5 * 2
2 + 4 * 47 / 8 - 5 * 2
2 + 188 / 8 – 5 * 2
2 + 23.5 – 5 * 2
2 + 23.5 – 10
Example: Order of Operation

2 + 4 * 47 / 2 ** 3 - 5 * (1 – 3 + 4)
2 + 4 * 47 / 2 ** 3 - 5 * (-2 + 4)
2 + 4 * 47 / 2 ** 3 - 5 * 2
2 + 4 * 47 / 8 - 5 * 2
2 + 188 / 8 – 5 * 2
2 + 23.5 – 5 * 2
2 + 23.5 – 10
25.5 – 10
Example: Order of Operation

\[ 2 + 4 \times 47 / 2 \times 3 - 5 \times (1 - 3 + 4) \]
\[ 2 + 4 \times 47 / 2 \times 3 - 5 \times (-2 + 4) \]
\[ 2 + 4 \times 47 / 2 \times 3 - 5 \times 2 \]
\[ 2 + 4 \times 47 / 8 - 5 \times 2 \]
\[ 2 + 188 / 8 - 5 \times 2 \]
\[ 2 + 23.5 - 5 \times 2 \]
\[ 2 + 23.5 - 10 \]
\[ 25.5 - 10 \]
\[ 15.5 \]
Exercise For Fun

See how many different answers you can get by adding parentheses to change the order of evaluation.

2 + 4 * 47 / 2 ** 3 - 5 * (1 - 3 + 4)
Converting Math Formulas to Programming Statements

- When converting mathematical expression to programming statement:
  - May need to add multiplication operators
    \[ r(b - 4ac) \text{ becomes } r \times (b - 4 \times a \times c) \]
  - May need to insert parentheses
    \[ \frac{Y - 47}{Z + 45.3} \text{ becomes } \frac{(Y - 47)}{(Z + 45.3)} \]
Formatting floating point

\[ a = \frac{10000}{6} \]

\[ b = \text{format} \left( a, '{:.2f}' \right) \quad \text{# format as a 2 digit float} \]

\[ c = \text{format} \left( a, '{:.5f}' \right) \quad \text{# format as a 5 digit float} \]

\[ d = \text{format} \left( a, '{,.5f}' \right) \quad \text{# format as a 5 digit float + comma separators} \]

\[ e = \text{format} \left( a, '{20,.2f}' \right) \quad \text{# format as a 2 digit float + commas + 20} \]

\text{# character min field width}
Formatting Numbers

• Can format display of numbers on screen using built-in `format` function
  – Two arguments:
    • Numeric value to be formatted
    • Format specifier
  – Returns string containing formatted number
  – Format specifier typically includes precision and data type
    • Can be used to indicate comma separators, and the minimum field width used to display the value
Review: Built-in functions

What are the argument types? Returns a value?

– *print*    print arguments to screen
– *input*   read string from keyboard
– *int*   convert argument to integer
– *float*   convert argument to floating point
– *str*   convert argument to string
– *type*   return type of argument
– *format*   convert argument to formatted string
Interesting Properties of Functions

- *print* can have an arbitrary number of arguments of different types (int, float, str)
- *print* does not return a value
- *print* takes keyword parameters (sep = ‘ ‘)
- *int, float, str* take one argument of different types (int, float, str)
- *type* returns a value that has a type that is neither int, float, or str

.....
Getting Help On Functions In IDLE

• Type `function_name()` and see flyover
• From toolbar: Help -> Python Docs
  – Library References
• Try it out and experiment
• You can use Google