Floating Point Numbers (float)

1. An approximation of Real numbers in math – Why?
    a. 10/3 = 1.3333…
    b. Computer can't store infinite number of threes
    c. Look at 10/3
    d. .1 + .1 + .1 ≠ .3

2. Floating Point Literals: 12.3 123e-1 4.3E7 -2.87e-3 see type(exp)

3. Precision of FP Numbers
    a. 2 ** 500 vs. 2.0 ** 500

Complex Numbers (complex)

1. Real part and Imaginary part. $j = \sqrt{-1}$

2. Complex Literals: see type(exp)

9 + 3j → (9+3j)	19 - 8j → (19-8j)	4j → 4j	1j → 1j	j → Error

3. Complex Expressions:

1j ** 2 → (-1+0j)	3j * 3j → (-9+0j)	3j ** 2 → (-9+0j)	6j * 2 → 12j

4. Extracting Real and Imaginary parts:

(36 -12j).real → 36.0	(3.2 + 2j).imag → 2.0

Types of Expressions

1. If either argument is complex, the expression is **complex:**
   9j * 10 (25+50j)/5

2. If either argument is Floating Point, expression is **Floating Point**

   2 * 3.1 → 6.2
   10.0 / 4 → 2.0
   10.0 // 4.0 → 2.0 Integer division, but operands are FP

3. Both must be Integers, expression is **integer**
   35 // 12 → 2 truncated 2.91666
   10 / 4 → 2.5 Special Case, / is FP division

Casting -Manually Converting Types

1. float(19) → 19.0
2. int(19.8) → 19
3. complex(12.34) → 12.34+0j
4. int(4j) → error
5. float(3j) → error

Strings

1. String Literals:
   a. Single Quote: 'kml'
   b. Double Quote: "kml"
   c. Triple Quote: '''kml''' or """kml"""

2. Special characters:
   a. New line \n
   b. Tab \t
   c. Backslash \\
   d. Single quote \'
   e. Double quote \"

3. See 01-08-stringLiterals.py

Section 1.5.2, Naming Objects (Miller 3rd ed)

1. Naming objects (identifiers)
    a. Rules:
        i. Can contain **a-z**, **A-Z**, **0-9**, or **_**
        ii. Cannot start with a number
        iii. Case sensitive

    b. Recommendations/conventions
        i. Use meaningful names (See Programming Style Guide/Class Website)
        ii. Don't start names with underscore