Questions about the lab?

General comments about the lab

- Make sure you are answering the question asked and answering them completely/accurately
  - Decimal points
- When I ask you to **predict** I mean it. I won't mark you off for being wrong so don't fake your "prediction"
- Try to be as clear as possible in your written answers

General comments about the lab

- How many base mathematical operators did we study in lab yesterday?
- Which one was the most confusing to you?

Operators

- addition : +
- subtraction: –
- multiplication: *
- division
  - quotient: /
  - Integer quotient: //
  - remainder: %
- exponentiation: **

New material (and some review):

- Literals
- Operator precedence
- Variables
  - naming conventions
  - assignment statement
- Constants
- Data types
  - common types; type conversion
- Using Python IDLE

Literals

- Literal is a programming notation for a fixed value.
- For example, 123 is a fixed value, an integer
  - it would be “weird” if the symbol 123’s value could change to be 3.14!

Number literals

- By default, IDLE assumes that anything without a decimal is an integer.

```
> 2
2
>  -5
-5
```

- By default, IDLE assumes that anything with a decimal is a floating point number.

```
>  3.5
3.5
>  -4.0
-4.0
```

Operator Order

- The default evaluation order is
  - Exponents **
  - Negation –
  - Multiplication and division *, /, //, %
  - Addition and subtraction, +, -
- The default order can be changed
  - By using parenthesis
  - (3 + 4) * 2 versus 3 + 4 * 2

Math Operator Order Exercise

- What is the result of

  2 + 3 * 4 + 5
- Where would you add parentheses to make it clear what is happening first
- How do you change it so that 2 + 3 happens first?
- How do you change it so that it multiplies the result of 2 + 3 and the result of 4 + 5?

But we don't often work with JUST literals…

- Normally when we do calculations we want to store the calculations to use later:
  - To do this we need to store values associated with a name (a variable)
- A variable is a name we designate to represent "something" in our program
- We use names to make our program more readable, so that the "something" is easily understood

Python Name Conventions

- must begin with a letter or _
  - `ab123` is OK, but `123ABC` is not.
- may contain letters, digits, and underscores
  - `this_is_an_identifier_123`
- may be of any length
- upper and lower case letters are different
  - `lengthOfRope` is not `lengthofrope`
- names starting with _ have special meaning. Be careful

Variable Objects

- Python maintains the following for every variable:
  - variable's name
  - variable's value
  - variable type
- A variable is created when a value is assigned the first time. It associates a name and a value
- subsequent assignments update the associated value.
- we say name references value
- A variable's type depends on what is assigned.

$X = 7$ ⟶

Name	Value
X	7

When = Doesn't Mean Equal

- It is most confusing at first to see the following kind of expression:
  - myInt = myInt + 7
- You don't have to be a math genius to figure out something is wrong there.
- What's wrong is that = doesn't mean equal

= is assignment

- In many computer languages, = means assignment.
  - myInt = myInt + 7
  - lhs = rhs
- What “assignment” means is:
  - evaluate the expression on the rhs of the =
  - take the resulting value and associate it with the name on the lhs (or copy the value into the variable on the left)

More Assignment

- Example: x = 2 + 3 * 5
  - evaluate expression (2+3*5): 17
  - change the value of x to reference 17
- Example (y has value 2): y = y + 3
  - evaluate expression (y+3): 5
  - change the value of y to reference 5

Constants

- We can now introduce constants
  - Written in ALL_CAPS_STYLING
  - Holds values that don't change throughout your program
- Why would we want to use constants?

Variables and Types

- Python does not require you to pre-define the type of a variable
- What type a variable holds **can** change
  - Unlike other programming languages…
- However, once a variable has a value it's type matters a lot!
  - Is x+3 legal?
- Thus proper naming is important!

Examples

- For example, you can't use the + operator with a string and an int
  - >>> "Sergey"+42
  - Python tries to convert one operand to the other operand's type
- What about the + operator with a float and an int?
  - >>> 4.1 + 3
  - >>> 3 + 4.1

Python "Types"

- integers: **5**
- floats: **1.2**
- Booleans: **True (note the capital)**
  - Boolean named after Mathematician George Boole
- strings: "anything" or 'something'
- lists: [,]: ['a',1,1.3]
- others we will see

Use of quotation marks with String type

- Python allows the use of either
  - Single quotes

    'This is a single quote sentence'
  - Double quotes

    "This is a double quote sentence"
  - Triple quotes

    """ These are most often used when writing comments that span over several lines."""

What is a Type?

- A type in Python essentially defines two things:
  - Internal structure of the data (what it contains)
  - Kinds of operations you can perform
- Different ***methods*** are associated with different types of data
  - >>> abc.capitalize()
    - Method you can call on strings, but not integers
    - Have you seen something like this before?

Converting Types

- A character ‘1’ is not an integer 1.
- You need to convert the value returned by the `input()` command (characters) into an integer
  - >>> inputString = “123”
  - >>> inputInteger = int(inputString)

Type Conversion

- int(someVar) converts to an integer
- float(someVar) converts to a float
- str(someVar) converts to a string
- should check out what works:
  - int(2.1) → 2, int('2') → 2, but int('2.1') fails
  - float(2) → 2.0, float('2.0') → 2.0, float('2') → 2.0, float(2.0) → 2.0
  - str(2) → '2', str(2.0) → '2.0', str('a') → 'a'

Knowing the type

- Since the type of data stored in a variable can change, Python let's us ask what the current type is:

```
type(variableName)
```

Using Python IDLE

- Interpreter
  - Has >>> prompt
  - Using numerical literals
  - Using variables and formulas
  - Try things in here!

- Programming area
  - Go to “File”, then “New File”
  - To write and save code into programs