Name:_____ **Section:**_____

Objectives

- Explain how Python uses types to characterize information.
- Use expressions to calculate mathematical quantities.
- Understand how operators work for different data types.

Python Family Values

At the hardware level, computers know only bits (0s and 1s) and bytes (collections of eight bits). Clearly for most purposes we require a much higher level of expression—words, images, programs. To create these, we have to interpret the raw bytes according to certain rules.

For instance, consider the byte of data written as □□■□■■□■ or 00101101. We can interpret this numerically by adding the following values together:

□	□	■	□	■	■	□	■
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0×2^7 +	0×2^6 +	1×2^5 +	0×2^4 +	1×2^3 +	1×2^2 +	0×2^1 +	1×2^0
0	+ 0	+ 32	+ 0	+ 8	+ 4	+ 0	+ 1 = 45

So 45 is the *integer* value of this byte. If we want to interpret the byte as a character (part of a string), then we use a character table. (These have hundreds or thousands of entries; we show a portion.)

Decimal Value	Character	Decimal Value	Character
40	'('	51	'3'
41	')'	52	'4'
42	'*'	53	'5'
43	'+'	54	'6'
44	','	55	'7'
45	'-'	56	'8'
46	'.'	57	'9'
47	'/'	58	'0'
48	'0'	...	...
49	'1'	65	'A'
50	'2'	66	'B'

The *data type* is what tells Python *which* rule to apply to the value, and thus whether you get an integer, a floating-point (scientific-notation) number, a character, a command, and so forth.

Integers are the whole numbers, positive and negative: …, –4, –3, –2, –1, 0, +1, +2, +3, …. One natural extension of integers are the *rational numbers*, numbers you can write as proportions or fractions: ½ ⅓ ⅘ $^{1,000}/_{1,001}$ $^{8}/_{5}$. If you think about it, decimal numbers like you've seen on a calculator are rational numbers written with respect to powers of 10:

¼ = 0.25 = $^{25}/_{100}$ ⅓ = 0.3 ≈ 0.33333 = $^{33,333}/_{100,000}$

Floating-point numbers are rational numbers written with respect to powers of ten, including scientific notation (*e.g.*, 1e5 = 10000.0). Finally, Python also supports **strings**, as you have seen. These are the three basic data types of Python.

Operators

A partial table of Python operators follows, with the name of each operator.

Operator	Numerical Operation	String Operation
+	addition	concatenation
-	subtraction	—
*	multiplication	repetition
/	division	—
**	exponent	—
%	modulus	—
=	assignment	assignment
//	floor division	—

Generally these behave as you would expect from using a graphing calculator. Parentheses may be useful to clarify the order of operations.

(1 + x) * 5 = 5 + 5 * x 1 + x * 5 = 1 + 5 * x

Submission

1. Finish the following exercise on paper, hand it in to Instructor at the end of today's lab session
2. Complete lab02.ipynb on Jupyter Notebook, save it, rename it to lab02_[your university ID].ipynb, and send it to Instructor by email (zliao@zju.edu.cn)

Exercise

1. Give the basic data type (int, float, str) for each of the following expressions.

Value	Data Type	Value	Data Type
'0'	_____	'2'	_____
1.	_____	2	_____
4e10	_____	int('45')	_____
1 / 4	_____	'0' * 3	_____

2. Write the following expressions as Python code. *a*, *b*, *x* are separate variables.

   a. $-a^b$

   b. $\frac{a}{b} + ab$

   c. $1 + \frac{x}{3} + \frac{a+b}{x}$

3. Why doesn't
   user-name = 'catherine'
   work as a possible variable name? (Think about the form of the variable name.)