Introduction to Python I

In this course, you will work in teams of 3–4 students to learn new concepts. The first activity will introduce you to the process. Next we'll take a first look at programmable internal-state machines and arithmetic expressions.

Meta Activity: Team Roles

Decide who will be what role for today; we will rotate the roles each week. If you have only three people, one should have two roles. If you have five people, two may share the same role.

Manager:	Helen Hu	keeps track of time, all voices are heard
Presenter:	Clif Kussmaul	asks questions, gives the team's answers
Recorder:	Chris Mayfield	quality control and consensus building
Reflector:	Aman Yadav	team dynamics, suggest improvements

Questions (15 min)   Start time:

1. What is the difference between **bold** and *italics* on the role cards?

The bold points describe what the responsibilities are. Examples of what that person would say are in italics.

2. Manager: invite each person to explain their role to the team. Recorder: take notes of the discussion by writing down key phrases next to the table above.

3. What responsibilities do two or more roles have in common?

Both the presenter and the recorder help the team reach consensus. The manager and reflector both monitor how the team is working.

4. For each role, give an example of how someone observing your team would know that a person is <u>not</u> doing their job well.

- Manager: The team is constantly getting behind.

- Presenter: The student doesn't know what to say.

- Recorder: Some team members aren't taking good notes.

- Reflector: The student never comments on team dynamics.

Model 1   Programmable Internal-State Machines

In this course, you will learn how to write instructions for the *Python machine* alongside the concepts they embody. Let's start with our first concept: the *programmable internal-state machine*.

Questions (10 min)   **Start time:**

5. What do you think of when you hear the term *machine*?

> Answers may vary. I would say a machine represents different components working together to perform a function.

6. What types of machines are there in the world, can you come up with categories or explain how they may be different? (**Hint:** Think an old coffee machine vs. an espresso machine that remembers to brew you a fresh cup of coffee every morning)

> Answers may vary. I would categorize machines in two categories: purely mechanical and machines that have internal memory.

7. What do you think of when you hear the term *programmable machine*? (**Hint:** Think basic instructions)

8. What do you think of when you hear the term *internal-state machine*? (**Hint:** Think variables)

> The Python machine is a broad term that encompasses the Python application (which you will install on your computer) and certain aspects of the operating system (MAC, Windows, Linux) and the underlying hardware of your computer that are relevant for programming in Python.
>
> The *Python machine* is a *programmable internal-state machine*. The name comes from the fact that it performs two fundamental functions: (1) it interprets the instructions in your program (related to programmable) and (2) it remembers information for later usage (related to internal-state). A *programmable machine* is able to perform some basic instructions (addition, input, output, etc.), which can be combined in different ways by programs to implement different functionality (ergo, the name programmable). A program can direct the internal-state machine to remember information for later usage via *variables*.

Model 2   Arithmetic Expressions

An instruction is represented in Python as an *expression*. A line in a Python program represents a *statement*. A statement can be made up of one or multiple expressions. There are two fundamentally different types of basic expressions: *function-call expressions* and *operator expressions*. We start with *arithmetic operator expressions*.

Questions (10 min)   Start time:

9. In the "Result" column, write what value you expect will be the result. If there are any lines you are not confident about, place an asterisk next to your answer:

Python code	Result
2 + 3	5
3 * 4 + 2	14
3 * 4 + 2.0	14.0
3 * (4 + 2)	18
5 / 10	0.5
5 / 10.0	0.5
5 / 9	0.5555556
2 ** 4	16

10. What does the ** operator do?

It raises a number to a power.

11. For addition and multiplication to produce an output with a decimal value, what type of number must be part of the input? Provide justification for your team's answer.

At least one of the numbers must have a decimal value. For example, 3 * 4 + 2 is the integer value 14, but 3 * 4 + 2.0 is the decimal value 14.0.

12. Does division follow the same rule as in #11? Provide justification for your team's answer.

No; when dividing integers, the result is always a decimal number. The same is true even when there is no remainder, i.e., 8 / 4 is 2.0.

Questions (15 min)   **Start time:**

13. For each operator expression, identify the symbol of the operator and describe the type of numerical result.

Table A			Table B			Table C		
9 / 4	*evaluates to*	2.25	9 // 4	*evaluates to*	2	9 % 4	*evaluates to*	1
10 / 4	*evaluates to*	2.5	10 // 4	*evaluates to*	2	10 % 4	*evaluates to*	2
11 / 4	*evaluates to*	2.75	11 // 4	*evaluates to*	2	11 % 4	*evaluates to*	3
12 / 4	*evaluates to*	3.0	12 // 4	*evaluates to*	3	12 % 4	*evaluates to*	0
13 / 4	*evaluates to*	3.25	13 // 4	*evaluates to*	3	13 % 4	*evaluates to*	1
14 / 4	*evaluates to*	3.5	14 // 4	*evaluates to*	3	14 % 4	*evaluates to*	2
15 / 4	*evaluates to*	3.75	15 // 4	*evaluates to*	3	15 % 4	*evaluates to*	3
16 / 4	*evaluates to*	4.0	16 // 4	*evaluates to*	4	16 % 4	*evaluates to*	0

/ decimal   // integer   % integer

Note that Python refers to decimal numbers as "floating-point" numbers.

14. If the result of the / operator were rounded to the nearest integer, would this be the same as the result of the // operator? Explain how the results in Table A compare to Table B.

No, the pattern is off by two rows. The 0.75 values would round up, but in the second table they round down. (0.5 values might round up or down depending on rounding rules.)

15.   If the table included more rows, list all numbers // 4 would evaluate to 2 and all the numbers // 4 would evaluate to 4.

8, 9, 10, and 11 evaluate to 2.
16, 17, 18, and 19 evaluate to 4.

16.   Based on the results of Table C, propose another number % 4 evaluates to 0, and explain what all these numbers have in common.

Other numbers include 0, 4, 8, 20, 24. All of these numbers are multiples of four.

17. Consider the expressions in Table C that evaluate to 1. How do the left *operands* in these expressions (i.e., 9, 13) differ from those that evaluate to 0?

They each differ by one; they are one higher than a multiple of four.

18. Describe the reason for the repeated sequence of numbers (0, 1, 2, 3) for the result of % 4.

The difference (remainder) increases by one until the number is exactly divisible by 4.

19. Recall how you learned to do long division in elementary school. Finish solving for $79 \div 5$ below. Which part of the answer is 79 // 5, and which part is 79 % 5?

We first bring the 9 down, then 5 goes into 29 five times, and so we subtract 25. The final answer is 15 remainder 4. So 79 // 5 is 15, and 79 % 5 is 4.

20. Imagine that you are given candy mints to divide evenly among your team members.

  a) If your team receives 11 mints, how many mints would each student get, and how many are left over? Write a Python expression to compute each result.

  11 / 3 is 3 and 11 % 3 is 2   or   11/ 4 is 2 and 11 % 4 is 3

  b) If your team receives 2 mints, how many mints would each student get, and how many are left over? Write a Python expression to computes this result.

  2 / 3 is 0 and 2 % 3 is 2   or   2 / 4 is 0   and   2 % 4 is 2

21. Python has three division operators: "floor division", "remainder", and "true division". Which operator (symbol) corresponds to each name?

// is floor division, because it throws away the decimal place (i.e., it "floors" the result by rounding towards negative infinity). % is the remainder operator, which is sometimes called the modulo operator. / is true division, because it gives you the mathematically correct answer.

Model 3   Variables

In programming, variables are the way to save information for later usage in the internal-state of the program. An **assignment statement** saves a value to a **variable**. The variable "is set to" the value after the "=" **operator**. For example:

```
mass = 10
```

Consequently, when a variable appears in an expression or statement, the value associated with the variable is **accessed** or **read**. For example:

```
print(mass)
```

In this example, the value of **mass** is first **accessed/read** and then printed by the **print statement**.

Demonstrate and explain the following lines: x = 10, `print(x)`, x + 1, `print(x)`, x = x + 1, and `print(x)`.

Questions (15 min)   Start time:

Consider the code examples in the table below and indicate the output:

Python code	Output
x = 12	
y = 2 + 5	
print(x,y)	12 7
x - 1	
print(x,y)	12 7
x = y	
print(x,y)	7 7
y = y + 1	
print(x,y)	7 8

22. Pick one assignment statement from the table above, and identify the following:

a) the variable being assigned   x

b) the assignment operator   =

c) the value of the variable immediately after the assignment   12

23. Show how the Python Machine interprets and executes the statements above:

Interpreter	Basic Instructions Pad	State
Assignment	x=12	x:12
Addition	5+2=7 (value not saved)	x:12
Assignment	y=7	x:12	y:7
Read	Replace x with 12	x:12	y:7
Read	Replace y with 7	x:12	y:7
Function: print	print(12,7)	x:12	y:7
Read	Replace x with 12	x:12	y:7
Subtraction	12-1=11 (value not saved)	x:12	y:7
Read	Replace x with 12	x:12	y:7
Read	Replace y with 7	x:12	y:7
Function: print	print(12,7)	x:12	y:7
Read	Replace y with 7	x:12	y:7
Assignment	x = 7	x:7	y:7
Read	Replace x with 7	x:7	y:7
Read	Replace y with 7	x:7	y:7
Function: print	print(7,7)	x:7	y:7
Read	Replace y with 7	x:7	y:7
Addition	7+1=8 (value not saved)	x:7	y:7
Assignment	y = 8	x:7	y:8
Read	Replace x with 7	x:7	y:8
Read	Replace y with 8	x:7	y:8
Function: print	print(7,8)	x:7	y:8

24. Is the way of interpreting and executing code used in #23 closer to how *ActiveCode* 1 or *CodeLens* 2 can be used in the book? Explain your answer. (See the back of the handout for a picture of each)

CodeLens because ActiveCode only allows you to run the code. CodeLens allows you to control the step by step execution of a program and shows you the values of all variables as they are created and modified.

25. After the successful execution of an assignment statement, how can you confirm the value of this variable?

You can print it to see its current value.

26. Indicate whether each statement below is true or false.

a) A variables can be set to different values throughout a program. true

b) A variables can store multiple values at the same time. false

c) The assignment operator works in Python just as it works in math. false

27. Write a line of Python code to assign the current value of `mass` to the variable `temp`. Show output that confirms that you have done this correctly, and explain the code.

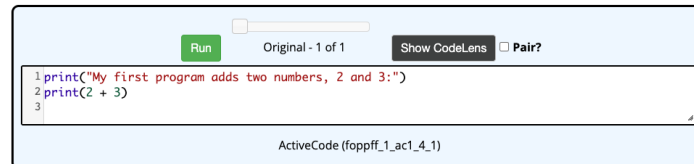The correct line is `temp = mass`. To verify the result, simply type `print`(temp). The assignment operation reads from right to left.
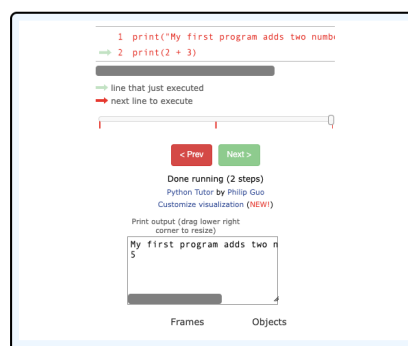

Figure 1: ActiveCode

Figure 2: CodeLens