Introduction to Python I

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 (5-10 mins) Start time:

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

2. 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)

3. What do you think of when you hear the term ***programmable machine***? (**Hint:** Think basic instructions)
What do you think of when you hear the term ***internal-state machine***? (**Hint:** Think 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 mins) Start time:

4. 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`	
`3 * 4 + 2`	
`3 * 4 + 2.0`	
`3 * (4 + 2)`	
`5 / 10`	
`5 / 10.0`	
`5 / 9`	
`2 ** 4`	

5. What does the ** operator do?

6. 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.

7. Does division follow the same rule as in #6? Provide justification for your team's answer.

Questions (15 min) Start time:

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

Table A

`9 / 4`	*evaluates to*	2.25
`10 / 4`	*evaluates to*	2.5
`11 / 4`	*evaluates to*	2.75
`12 / 4`	*evaluates to*	3.0
`13 / 4`	*evaluates to*	3.25
`14 / 4`	*evaluates to*	3.5
`15 / 4`	*evaluates to*	3.75
`16 / 4`	*evaluates to*	4.0

Table B

`9 // 4`	*evaluates to*	2
`10 // 4`	*evaluates to*	2
`11 // 4`	*evaluates to*	2
`12 // 4`	*evaluates to*	3
`13 // 4`	*evaluates to*	3
`14 // 4`	*evaluates to*	3
`15 // 4`	*evaluates to*	3
`16 // 4`	*evaluates to*	4

Table C

`9 % 4`	*evaluates to*	1
`10 % 4`	*evaluates to*	2
`11 % 4`	*evaluates to*	3
`12 % 4`	*evaluates to*	0
`13 % 4`	*evaluates to*	1
`14 % 4`	*evaluates to*	2
`15 % 4`	*evaluates to*	3
`16 % 4`	*evaluates to*	0

9. 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.

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

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

12. 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?

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

14. 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`?

15. 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.

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.

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

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***.

Questions (15 min) Start time:

Consider the code examples in the table below, evaluate them and indicate the output:

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

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

a) the variable being assigned

b) the assignment operator

c) the value of the variable immediately after the assignment

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

Interpreter	Basic Instructions Pad	State	

19. Is the way of interpreting and executing code used in #18 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)

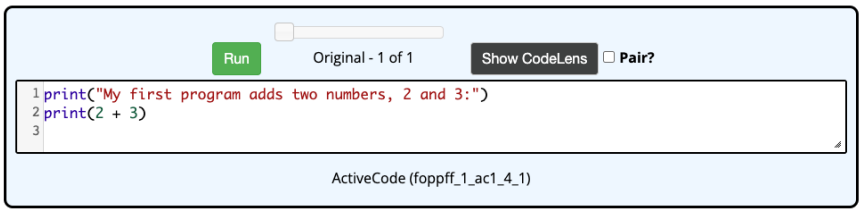

Figure 1: ActiveCode

Figure 2: CodeLens

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

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

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

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

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

22. 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.