



NOSTARCH.COM/PFK

For bulk orders, please contact us at
sales@nostarch.com.

| | | | |
|----------|----------------------|--------------|---------------|
| Teacher: | | Date/Period: | |
| Subject: | Python Programming | Class: | |
| Topic: | #1 - Getting Started | Duration: | Up to 50 min. |

| | |
|-------------|--|
| Objectives: | <ul style="list-style-type: none"> ● Install the latest Python3 distribution. ● Recognize the difference between the Python console and the shell (IDLE) and know when to use one over the other. ● Create a simple program. Run it from the command line and from within the shell. ● Use the shell (or console) as a simple calculator. ● Describe what a program is and what a programming language is. |
| Materials: | <ul style="list-style-type: none"> ● Python 3 installer (or link to download the installer) ● A short Python program to run (as a suggestion, one of the turtle examples from Chapter 11: http://jasonrbriggs.com/python-for-kids/code.html#Chapter-11/) |
| Activities: | <ol style="list-style-type: none"> 1. Download and install the latest version of Python. (Depending on the environment, you may want to shortcut this process by predownloading or making the installer accessible from the school network.) [5–10 min.] 2. Start up the Python console, and explain what the prompt is. Start up the Python shell and compare. Try out basic print statements in both. [5–10 min.] 3. Open a Python program and run it from within the shell. [5 min.] 4. Create a new window in the shell, and enter a simple program (for example, Hello World). Save and run the program. [5 min.] 5. Discussion: How are programming languages instructions to the computer? What does it mean for something to be human-readable versus computer-readable? What's the difference between scripting languages (such as Python) and compiled languages (such as C or C++)? Which would be used to create most of the programs seen on a day-to-day basis? [10 min.] 6. Open the shell again. Try entering some basic calculations and seeing the results. Demonstrate the different operators (add +, subtract -, multiply *, and divide /), and show how the use of brackets affects the result of a calculation. [5–10 min.] |

| | |
|-------------|---|
| Notes: | <ul style="list-style-type: none">• Depending on your lab environment, have students download from the Python website, from a local cache, or from the school website. At the very least, the process should be reproducible at home.• Take careful note of the installation instructions on pages 5–10, specifically those on adding a shortcut to run the shell from the desktop on Mac and Windows, to avoid experiencing errors later. |
| References: | <i>Python for Kids</i> , Chapters 1 & 2 (up to page 19) |

| | | | |
|----------|-----------------------------|--------------|---------------|
| Teacher: | | Date/Period: | |
| Subject: | Python Programming | Class: | |
| Topic: | #2 Storing Things in Python | Duration: | Up to 60 min. |

| | |
|-------------|---|
| Objectives: | <ul style="list-style-type: none"> ● Understand how a variable stores things. Know the difference between a slot in memory used to hold an actual value and a variable used as a label to point at a value. ● Understand the difference between a number and a string. ● Understand the purpose of lists and maps. ● Understand the difference between a tuple and a list. ● Use a tuple with a string containing placeholders. |
| Activities: | <ol style="list-style-type: none"> 1. Briefly review the prior lesson: opening the shell and running a program. [5 min.] 2. Create a program to store a number in a variable, and then print out the number. Discussion: What do you use variables for? Why do programs need variables? [5-10 min.] 3. Try to create a variable containing a silly sentence. After the error message is displayed, discuss the difference between strings and numbers. Create different variables with strings and numbers. [5–10 min.] 4. Discussion: Are there other things you might want to store in variables? Introduce lists and maps. [5 min.] 5. Create variables with simple lists of things (students' favorite items, for example). Remove items from a list by their index. Discussion: If you stored a list of items in a variable, how would you remove a single item from the list? How do you remove that item if the list is a string? (You have to recreate the string.) [10–15 min.] 6. Create a tuple. Try to remove an element from the tuple. Discussion: Why use a tuple over a list? (Tuples are faster because their values can't change.) [10 min.] 7. Create a string with a couple of placeholders (%s). Use a tuple for the values when printing out the string. Discussion: What is Python doing here? Why are placeholders useful? [10 min.] |
| Notes: | N/A |
| References: | <i>Python for Kids</i> , Chapters 2 & 3 |

| | | | |
|----------|------------------------------|--------------|---------------|
| Teacher: | | Date/Period: | |
| Subject: | Python Programming | Class: | |
| Topic: | #3 - Drawing with the Turtle | Duration: | Up to 50 min. |

| | |
|-------------|---|
| Objectives: | <ul style="list-style-type: none"> • Understand how to import the <code>turtle</code> module and draw simple shapes. • Understand what a module is (as a unit of code). |
| Activities: | <ol style="list-style-type: none"> 1. Briefly review the prior lesson: variables, numbers and strings, lists and maps, and tuples. [5 min.] 2. Open the shell and import the <code>turtle</code> module. Discussion: What is a module? (Describe it as a small program that you can use inside your own programs.) [5 min.] 3. Draw a line with the turtle. Discussion: A line is made up of pixels—what is a pixel? Optional activity: Get a magnifying glass and have students find the edge of a pixel on the screen. [5–10 min.] 4. Turn the turtle 90 degrees and draw another line. Try turning the turtle 90 degrees to the right and 90 degrees to the left. Draw a zig-zag line. Discussion: What do degrees measure? Optional activity: Use a clock to illustrate the major degrees (45, 90, 135, etc.). [10 min.] 5. Try drawing a square at different sizes. Reset and then try drawing a square on an angle. Try using the <code>up</code> and <code>down</code> commands to start and stop drawing. Offer a period of free experimentation. [15–20 min.] |
| Notes: | <ul style="list-style-type: none"> • If your students haven't set up Python according to the instructions in Chapter 1, they may experience problems when using the <code>turtle</code> module, such as the shell seeming to hang. • Depending on the math level of the students, it may be a good idea to cover a lesson on degrees in parallel to this lesson (for an example, see http://www.homeschoolmath.net/teaching/g/measure_angles.php/). • If the students want to start over with their drawings, there are two options: use the <code>reset</code> function, or close the window and start again. |
| References: | <i>Python for Kids</i> , Chapter 4 |

| | | | |
|----------|-------------------------|--------------|---------------|
| Teacher: | | Date/Period: | |
| Subject: | Python Programming | Class: | |
| Topic: | #4 - Control Statements | Duration: | Up to 60 min. |

| | |
|-------------|--|
| Objectives: | <ul style="list-style-type: none"> ● Understand what blocks of code are in Python. ● Understand what an <code>if</code>-statement is and how to use it. ● Understand what a function is. |
| Activities: | <ol style="list-style-type: none"> 1. Briefly review the prior lesson: the <code>turtle</code> module and a module as a unit of code that you can use in your programs. [5 min.] 2. Discussion: What is a block of code in Python? How does Python identify a block of code? Why do you need blocks of code? [5 min.] 3. Discussion: <code>if</code> statements are used to control which block of code to run. How do you create an <code>if</code> statement? [5 min.] 4. Open the shell, create a new editor window, and then type a variable <code>age</code> with the student's age as the value. Create a simple <code>if</code> statement that prints one statement if the variable is greater than a specified value and another if it is not. (Get creative with silly messages here.) Discussion: What do you think the code is going to do when you run it? Why? [5–10 min.] 5. Run the code once to display the message. Then change the code so that the other statement is printed. Review how the code works. [5 min.] 6. Discussion: Break down the code, describing the keywords (<code>if</code> and <code>else</code>), the condition (for example, <code>age > 11</code>), and the statements forming the two code blocks. [5 min.] 7. Have students alter the conditions. Cover each conditional symbol (equal <code>==</code>, not equal <code>!=</code>, greater than <code>></code>, less than <code><</code>, greater than or equal to <code>>=</code>, and less than or equal to <code><=</code>). [5–10 min.] 8. Discussion: What might you use an <code>if</code> statement for (given the examples so far)? Briefly describe a function as a small unit of code that usually returns a value. Give the students an example of using the <code>int</code> and <code>input</code> functions. Have students type their age and save it as a variable. Update the code to use the <code>int</code> and <code>input</code> functions, and change the print messages. Run the new code, and then have students swap computers with each other. [15 min.] |
| Notes: | <ul style="list-style-type: none"> ● Remind students that they've been using functions since the second lesson (printing out a number using the <code>print</code> function, for example). |

| | |
|-------------|------------------------------------|
| | |
| References: | <i>Python for Kids</i> , Chapter 5 |

| | | | |
|----------|-------------------------|--------------|---------------|
| Teacher: | | Date/Period: | |
| Subject: | Python Programming | Class: | |
| Topic: | #5 - Control Statements | Duration: | Up to 55 min. |

| | |
|-------------|---|
| Objectives: | <ul style="list-style-type: none"> ● Understand how to repeat code (loops). ● Use basic <code>range</code> and <code>list</code> functions. |
| Activities: | <ol style="list-style-type: none"> 1. Discussion: Briefly review the prior lesson: <code>if</code> statements, blocks of code, and functions. [5 min.] 2. Discussion: Why would you want to repeat a block of code in a program? (As a starting point for the discussion, consider a character's movements in a game—if you push the right arrow, the character needs to walk to the right. How do you make him move across the screen without repetition?). [5 min.] 3. Open the shell and run <code>range(10)</code> Then run <code>list(range(10))</code>. Brief discussion: What happens when you run the first function (perhaps discuss iterators) versus when you run the second function? Variation: Run <code>list(range(5, 10))</code>. [5 min.] 4. Enter a simple <code>for</code> loop, for example: <pre>for x in range(10): print('Hello there')</pre> <p>Discussion: What is this code doing? [10 min]</p> 5. Now use a placeholder (<code>%</code>) in a string with a loop to print the value. Try the following code: <pre>for x in range(5, 10): print('Hello there %s' % x)</pre> <p>Discussion: What is this code doing? [10–15 min.]</p> 6. Demonstrate that the <code>for</code> loop will also work with lists of things (rather than just functions) by entering the following code: <pre>some_words = ['amazed', 'flabbergasted', 'flummoxed'] for x in some_words: print(x)</pre> <p>Discussion: What is this code doing? Is there a difference from the previous code where we used <code>range</code>? [10–15 min.]</p> |
| Notes: | N/A |

References:

[Python for Kids](#), Chapter 6

| | | | |
|----------|--------------------|--------------|---------------|
| Teacher: | | Date/Period: | |
| Subject: | Python Programming | Class: | |
| Topic: | #6 - Code Reuse | Duration: | Up to 55 min. |

| | |
|-------------|---|
| Objectives: | <ul style="list-style-type: none"> ● Understand how to create basic functions. ● Use the <code>time</code> module and the <code>sys</code> module. |
| Activities: | <ol style="list-style-type: none"> 1. Discussion: Briefly review the prior lesson: <code>for</code> loops and the <code>list</code> and <code>range</code> functions. [5 min.] 2. Discussion: Python has built-in functions (such as <code>print</code>, <code>list</code> and <code>range</code>), but you can also create your own functions. Why is this useful? (Functions are like mini programs. You can reuse code in different places in your programs.) [5 min.] 3. Open the shell and create a simple function for students to print their names, for example: <pre>>>> def simplefunc(): print('My name is Maximillian Shufflebottom')</pre> <p>Run the function. Discussion: Talk through the parts of a function (the <code>def</code> keyword, the function name, and the function body). [5–10 min.]</p> 4. Change the function to accept a parameter <code>name</code> and then print the name in the function body: <pre>>>> def simplefunc(name): print('My name is %s' % name)</pre> <p>Discussion: Talk about functions taking parameters. [5–10 min.]</p> 5. Now create a new function that returns a value, for example, the <code>savings</code> function in Chapter 7: <pre>>>> def savings(pocket_money, paper_route, spending): return pocket_money + paper_route - spending >>> print(savings(10, 10, 5)) 15</pre> <p>Discussion: What's happening here? (Mention that <code>return</code> is a keyword, similar to <code>def</code>.) [5–10 min.]</p> 6. Discussion: The students have already used the <code>turtle</code> module, but Python has a lot of different modules. Discuss the relationship between functions and modules (as groups of functions and variables) Show that they are both ways to reuse code. [5 min.] |

| | |
|-------------|---|
| | <p>7. Have the student import the <code>time</code> module and print the result of the <code>asctime</code> function. Then import the <code>sys</code> module and use the <code>readline</code> function of the <code>stdin</code> object (<code>sys.stdin.readline()</code>).</p> <p>Discussion: Check the students' understanding of what's happening here. (Focus on the fact that <code>stdin</code> is a variable in the <code>sys</code> module—don't go into objects at this point.) [10 min.]</p> |
| Notes: | <ul style="list-style-type: none">• The <code>time</code> and <code>sys</code> modules aren't the most interesting, but using them demonstrates that there are other modules besides the turtle. |
| References: | <i>Python for Kids</i> , Chapter 7 |