|  |
| --- |
| **The Basics Notes:**  **Introduction to Python**  In this course, you will learn the basics of Python and game design. At the end of the course, you will build your own Python games. The course is split into three parts that focus on different topics.    **Part I - Syntax and Basics**  Go on an adventure in the jungle to discover hidden treasures. You will use basic Python programming to navigate a character, avoid enemies, and collect treasure along the way.  **Part II - UI and Interaction**  You'll learn about the turtle tool, background color, x-y coordinates, keyboard interaction, and pen drawing.  **Part III - Game Design**  The third part will focus on putting it all together to build a game. You'll build several games on your own!  **Lessons**  Each lesson is made up of different modules:   * Videos * Descriptions of Python concepts and syntax * Puzzles that you solve by typing in code * Do-It-Yourself modules where you can showcase your creativity * A review section that goes over the concepts covered in the lesson * A quiz at the end of the lesson |
| **What Are Comments?**  Programmers often work on large teams editing a single base of code, and you may be working with others on your projects as well. You need to document your code and explain what you're doing so that anyone else who looks at your code understands what you're doing.  Python makes this easy for you.  **Single Line Comments**  In Python, you type the pound character "#" to start a new comment. Any text that you write on that line thereafter will be documentation for your code that will not affect how your program runs. Here's an example:    # Here are the commands to get to the treasure  forward() # Start moving forward one square  turn\_right() # Turn right so we can point down  forward() # Move forward again so we get further ahead  # After the previous step, we should be on the treasure!  Notice how the text after the "#" character is treated as documentation. This will not affect the commands for the forward, turn right, forward sequence. It doesn't matter where on the line the "#" characters start. Once this is entered, any text after it will not be run.  **Multi-Line Comments**  Sometimes you'll want more than one line of text to describe a section of code.  A multi-line command starts with **'''** and ends with **'''**.  Any text between the start and end marker will be ignored by Python.    '''  Here are the commands to get to the treasure. First we'll  start moving forward one square. Then we turn right so that  we point down. Then we move forward again so we get further  ahead. We'll be on the treasure after the last step!  '''  ​  forward()  turn\_right()  forward()  Single line comments are very common in code. Multi-line comments are used for more detailed documentation.  Comments also come in handy when you need to ignore some of your code to test things out or if you have alternate code that you don't want to run at the moment.  **Do it Yourself**  Try commenting and uncommenting the lines and adding your own.    print("Hello")  ​  # print("Hello there")  ​  **Note:**  The print command makes it easy to see the output in our example exercises.  Here's how it works:   |  |  | | --- | --- | | **Method** | **Description** | | print(value) | The print method will print the passed value in the play area on a single line. Each print method will start on a new line. | |
| **What Are Naming Conventions?**  In this lesson, you will learn about properly naming variables and functions in order to avoid confusion and provide structure to your code.  **Conventions**  In Python, we follow some rules when naming variables and functions.   |  |  | | --- | --- | | **Convention** | **Description** | | **Variable and function names** | Variable and function names start with lowercase letters. For example: color, move, forward. | | **snake\_case** | If a variable or function name has multiple words, make everything lowercase and separate words with underscores. For example: game\_over, the\_largest\_one, first\_name, last\_name.  This naming convention helps with understanding and debugging code. |   **Example 1**  These variables are not following the **snake\_case**naming convention. This won't generate an error, but they make your code harder to read.    # modify these names to be snakecase  shoeSize = 5  ShApEsIzE = 9  def GETshapeSIZE():  return ShApEsIzE  **Reserved Words**  Another important thing to keep in mind when naming your variables and functions is to not use **reserved words**.  Reserved words are identifiers that are built into the language, which means you cannot use them.  **Example 2**  You cannot use "**def**" as a variable name because "**def**" is a "**reserved word**" used to declare functions.    #this does not work  def = 4  print(def)  ​  Python does not allow the use of "def" as a variable name.  If you use reserved words, your code will not work. You need to use a valid variable name that isn't a reserved word.  Some common reserved words are:   |  |  | | --- | --- | | **Reserved Word** | **Use** | | def | Declare a function | | for | Create "for" loops | | if | Create "if" and "if-else" statements | | else | Create "if-else" statements | | while | Create "while" loops | | break | Break out of loops | | class | Declare a class definition |   **Do It Yourself**  Fix the code below so that it prints out all the variables. Make sure you follow **snake\_case** naming convention and do not use any **reserved words** as variable or function names!    def = 5  class = 8  CUPcAkE = 9  ​  print(def)  print(class)  print(CUPcAkE)  ​ |
| **Review**  Let's review what you've learned so far. Use this as a guide for the quiz coming up next!  **Syntax**  The rules by which Python understands the commands you type is called **syntax.**  **Commands**  The syntax for commands in Python is as follows:   * the command name * followed by parentheses   **Function Calls**  The commands we have used so far are **function calls**.  For example, the forward() command is making a function call to the function named **forward**.  Examples:   |  |  | | --- | --- | | **Syntax** | **Valid?** | | forward() | Valid | | forward(); | Valid | | forward | Invalid | | forward; | Invalid | | forward( | Invalid | | forward) | Invalid |   **Function Names**  Function names in Python are a type of **Identifier**.  Rules for function names (or identifiers):   * Cannot have spaces. * Must begin with a letter, or an underscore ( \_ ) * Can contain letters and numbers. * Cannot contain symbols such as exclamation marks (!), hyphens (-), periods (.), commas (,), etc. * Upper and lower case letters are not the same.   Examples:   |  |  | | --- | --- | | **Syntax** | **Valid?** | | forward 1() | Invalid because function names cannot have spaces | | 1forward() | Invalid because function names cannot start with a number | | oneforward() | Valid | | forward1() | Valid | | forward!!() | Invalid because function names cannot have symbols such as exclamation points | | ForWarD() | Technically valid but does not follow Python naming conventions (should be all lower case) |   **Comments**  Commenting is a way to write an explanation of your code. It helps others understand your code and add to it. Comments do not affect how your program runs.  **Single-Line Comments**  Python syntax for single-line comments is as follows:   * Type the pound character "#" to start a new comment. * Any text that you write on that line thereafter will be documentation for your code.   Example:  forward()      # this moves the character one step forward  turn\_right()    # this turns the character to the right  **Multi-Line Comments**  Sometimes you want more than one line of text to describe a section of code.  Python syntax for multi-line comments is as follows:   * Starts and ends with '''. * Any text between the start and end markers will be ignored.   Example:  '''  Here are the commands to get to the treasure. First, we'll  start moving forward one square. Then we turn right so that  we point down. Then we move forward again so we get further  ahead. We'll be on the treasure after the last step!  '''  Multi-line comments are used for more detailed documentation.  **Naming Conventions**  In Python, we follow some rules when naming variables and functions.   |  |  | | --- | --- | | **Convention** | **Description** | | **Variable and function names** | Variable and function names start with lowercase letters. For example: color, move, forward. | | **snake\_case** | If a variable or function name has multiple words, all words will be lowercase and separated by underscores. For example: game\_over, high\_score, first\_name, last\_name.  This naming convention helps with understanding and debugging code. | |
|  |
|  |
|  |
|  |