Control Flow: Conditional Statements

The ability to select what actions to perform based on circumstances we encounter is critical in real life and also in programming. A **conditional statement** (also called a **selection statement**) allows a program to do that. An *if*-statement instructs the shell to select between executing a suite or just skipping the suite; an *if-else* statement instructs the shell to select exactly one of two suites to execute; and an *if-elif* statement or an *if-elif-else* statement instructs the shell to select exactly one of two or more. This exercise focuses on using *if-else* statements to control the flow of execution.

**Part (a):** With a partner, play the game “What kind of animal were you in a prior lifetime?” After you have played it a couple of times, we will demo a Python program that automates the game.

**Part (b):** With your partner, download `game1.py`. Execute it a few times. Study it and be ready to answer the following questions about it.

1. What lines of code make up the first *if-else* statement in this program?

2. What lines of code make up the second *if-else* statement?

3. What lines of code are executed if the user types
   a. *Y* at the first prompt and *n* at the second?
   b. *y* at the first prompt and *y* at the second?
   c. *n* at the first prompt?

4. The `print('NOT DONE')` statement in this program is called a **print stub**. What purpose does the print stub serve?
Part (c): Finish implementing this program incrementally, as follows:

1. Replace the print stub with code that implements the True (YES) branch of the next question (Do you like cheese?) and uses a print stub on the False (NO) branch. **Run two tests of the new program before proceeding** – one that causes the program to print *You were a mouse!* and one that causes it to print *NOT DONE*.

2. Replace the print stub with code that implements the last question (Do you like to swim?). Run two tests of the new program – one that prints *You were a fish!* and one that prints *You were a cat!*

3. What advantages accrue to implementing the program in this fashion (i.e., incrementally)?

Part (d): Add error checking to your program in part (c). In other words, modify the program so that it plays the game as shown in the diagram provided that the user enters *Y, y, N, or n* at each prompt; but prints an error message and quits, if the user types any other input at any of the prompts. (Hint: The if-elif-else statement will be useful.)