Object Oriented Programming

What is Object Oriented Programming (OOP)?

1. Object-oriented paradigm entails the development of active program units called objects, each of which contain procedures describing how that object should respond to various stimuli.
2. Object-oriented programming is contrasted with procedural programming where a program is comprised of various procedures.
3. An object represents an entity in the real world that can be distinctly identified. For example, a student, a desk, a circle, a button, and even a loan can be viewed as objects.
4. Effectively, you can think objects as nouns which can be described with adjectives(states) and their actions can be described with verbs(behaviours)
5. An object has a unique identity, state and behaviour.
6. An object identity is like a person's social security number. Python automatically assigns each object a unique id for identifying the object at runtime.
7. An object's state (also known as its properties or attributes) is represented by variables, called data fields.
8. A circle object, for example has a data field radius which is a property that characterizes a circle. A rectangle object has the data field width and height, which are properties that characterize a rectangle.
9. Python uses methods to define an object's behavior (also known as its actions). Recall that methods are defined as functions.
10. You make an object perform an action by invoking a method on that object. A circle object can then invoke the getArea() method to return its area and the getPerimeter() method to return its perimeter.

Example 1

1. Situation : You are playing a game of hot and cold where you ask a player to guess a number between one and a hundred. First, you choose a number between one and a hundred which is hidden from the player. You will then ask your player to make a guess at what the number could be. If they guess it right then you congratulate them on their correct guess. If they didn't guess right, then you compare their guess with their previous guess, and if they are closer to the correct answer then you respond with "You are getting hotter" or if they aren't getting closer, you will respond with "You are getting colder." The game is repeated until the player gets it correct or until they have made 10 guesses whichever is first.
2. **Questions to ask yourself**:
   a. What are the steps to solving this problem?
   b. What are the objects used in solving this problem?
   c. What behaviours belong to what objects?

d. What attributes or property belong to what objects?

3. **Steps to solving this problem**:
   a. select_number_for_guessing() - Randomly selects a number for the player to guess
   b. prompt_player_to_guess() – Ask player to guess the correct number between 1 and 100. Player will then input their guess -guess();
   c. did_player_guess_correctly() – Determine whether player guessed correctly
   d. if player guess correctly then : congratulate_player() – Tell the player they won
   e. if player didn't guess correctly then:
      i. increment_number_of_guesses() – increase the number of guess the player has taken without getting it correct.
      ii. If the threshold for guess hasn't been exceed then display_hotter_colder() – tells the player whether they are getting hotter or colder then go back to b.
      iii. If the threshold for guessing has been exceeded then display_losing_message() – Tell the player they have lost
4. To come up with the steps, try asking yourself what's next if you were to actually do this in real life i.e. what actions do I take.

5. **Based on the steps to solving the problem what are the objects i.e. what entities are needed for the carrying out of this procedure?**
   a. You – We can give you a generic name like GameOperator
   b. The player – Who we can simply call Player

6. **What behaviours belong to GameOperator?**
   a. select_number_for_guessing()
   b. prompt_player_to_guess()
   c. did_player_guess_correctly()
   d. display_hotter_colder()
   e. display_losing_message()
   f. congratulate_player()

7. **What behaviours belong to Player?**
   a. guess()
   b. increment_number_of_guesses()

8. **What attributes belong to GameOperator?**
   a. target_number - the number the player is trying to guess
   b. THRESHOLD – a constant about the maximum number of guesses allowed

9. **What attributes belong to Player?**
   a. number_of_guesses – record the number of guesses the player made
   b. current_guess – stores the Players current guess

    c. previous_guess – stores the Player's previous guess
10. Implementation : https://replit.com/@ChesterGrant/hotcold#main.py
11. Why OOP?
    a. Object-oriented programming places data and operations that pertains to them together in an object which mirrors the real world, in which all objects are associated with both attributes and activities.
    b. Using objects improves software reusability and makes programs easier to develop and easier to maintain*.
    c. It is very easy to partition the work in a project based on objects
    d. The principle of data hiding helps the programmer to build secure programs which cannot be invaded by the code in other parts of the program
    e. Message passing techniques is used for communication between objects which makes the interface descriptions with external systems much simpler.

12. Why not OOP?
    a. The length of the programs developed using OOP language is much larger than procedural approach.
    b. Programmers need to have brilliant designing skills and programming skill along with proper planning because using OOP is little bit tricky.