Object Oriented Programming, Exercise 8

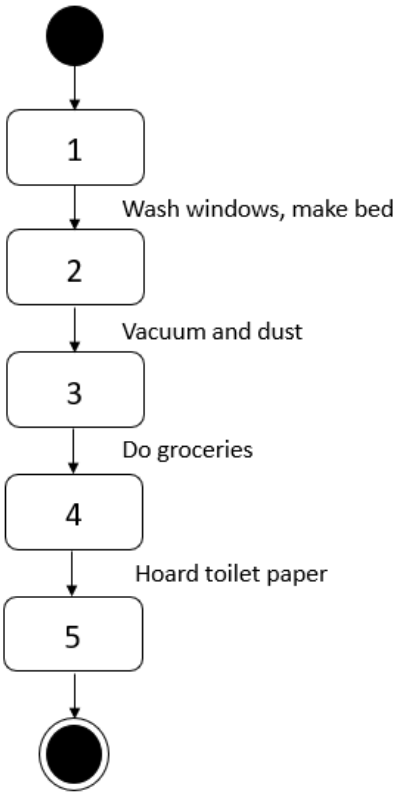
Topics: State diagrams and flow charts

Make a Git commit at least after every coding task.

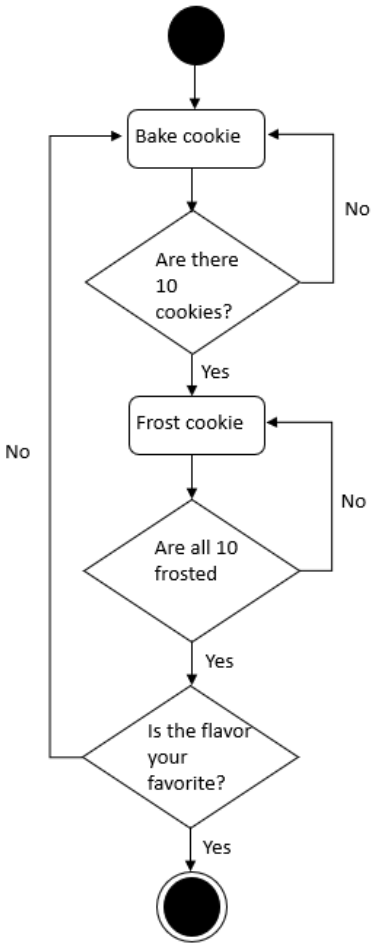
Code in Python3 and follow the style guide.

1. It is time to spring clean your house/flat. Your house/flat shall have floors, windows and surfaces that need cleaning and a bed that needs making. In addition, we shall prepare for the lockdown (fill the refrigerator and hoard toilet paper, as Finns usually do in crisis situations). **Implement the state diagram described below (next page).** So, code a class (House or Flat) and have the necessary data attributes, setters, getters, init and str method. Then make an instance of the class (in main function) and start cleaning. House/flat can have any number of windows, rooms, etc. (as you wish, decide some reasonable amount, it does not really matter here). Instance can have different states like this:
    - State 1: windows dirty, floors dirty, bed unmade, surfaces dusty, fridge empty and toilet paper running out.
    - State 2: like state 1 but windows are washed, and bed is made.
    - State 3: like state 2 but floors are vacuumed, and surfaces are dusted.
    - State 4: like state 3 but shopping is done so the fridge is full and there is enough toilet paper.
    - State 5: like state 4, but there is more than enough toilet paper.

    Do not overthink or "over implement" this task! You need a main function where you control the state transitions (using setters). You need the str-method to print the state of the object (see that output prints (=str method) are informative, and it is easy to see the state of the object). E.g. "washing the windows" is done with setter setting the state of the windows data attribute.



2. Change the diagram of Task 1 so that you ask the user, if he/she want to do a task or no (select 1 for yes and 2 for no). Change the code also. Use exception handling for verifying that user does input an integer and the input is either 1 or 2. See example from task 4 how to use a decision point (the diamond shaped element) in the diagram.
3. You got so excited about your polished house/flat that you want to clean your car and change the tires as well (and if you don't have a car, you are such a caring friend/parent/child that you use polish the car of your friend/parent/child). **Draw** a state diagram of that. Data attributes include e.g. tires (winter or summer tires), interiors vacuumed or dirty, car washed or dirty, tank full, empty or something in between (decides yourself). Use the same notation as used in task 1 (btw, it is drawn using power point). Do not use any colors or any funky elements in the diagram. (Code the diagram if you like, but this is not mandatory.)
4. After all the hard work of spring cleaning, it is time to bake some cookies. **Implement the following flow diagram.** Do not overdo/-think or "over implement" it, it is supposed to be simple! You will manage with the main function controlling the flow (and e.g. putting objects in a list etc.) and cookie class with setters, getters, init and str methods (and random flavor method). You can ask the favorite flavor from user (in main function). Bake and frost cookies one by one (let's assume you don't know at the beginning how many you need; however, for the sake of simplicity, we bake 10 (one by one)).



5. Go back to the alarm clock task of Exercise 2. What kind of state chart would describe your solution (or a possible solution)? Draw the diagram.
6. Continue with the "Countries and Capitals" quiz of Exercise 7. Give the user a chance to test his/her knowledge of countries based on their capital. Use exception handling when verifying user input.
7. Find some (freeware or free trial period) tools, which generate code from UML diagrams. Mention a few but try out at least one of them and analyze how the code is different when it is made by you and when it is generated by a tool.
8. Find out if any tools exist that can generate a UML diagram out of code. If yes, dig out more information and do an experiment (if you find freeware or free trial period tools).
9. Dig deeper in exceptions in Python.
    a. https://docs.python.org/3/library/exceptions.html
    b. https://www.programiz.com/python-programming/exceptions
    c. Code an example code for at least 5 different exceptions (try to select those that you understand and that are common, e.g. zero division). (2 points)