**Name**: Steve Hommy

**Pair: -**

**Amount of completed tasks: 9**

**Which tasks were left undone or incomplete: 0**

Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

Doing this exercise, I learned…

I am still wondering…

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

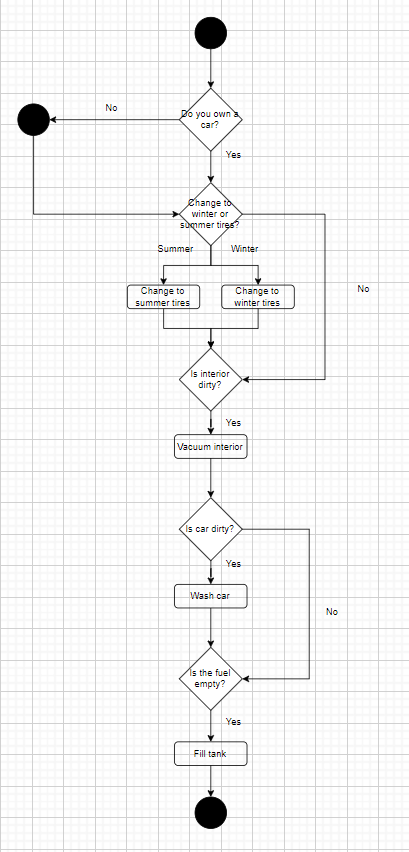
## Test report

Write the test report yourself to each coding task (task number, input/action, desired output and then the testing evidence (actual output)). Add rows if necessary. Include answers to theoretical questions and pseudocode to this return document as well in addition to code screen captures. Actual output can be a screen capture of the terminal showing the output.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Input / action** | **Desired output** | **Actual output (use red color if desired output != actual output)** |
| **1** | <Run Program> | Houses status:  Bedroom:  Window: Dirty  Floor: Dirty  Bed: Unmade  Surfaces: Dusty  Kitchen:  Window: Dirty  Floor: Dirty  Fridge: Empty  Surfaces: Dusty  Bathroom:  Surfaces: Dusty  Toilet paper: Running out  Washing the windows and making the bed  Vacuuming the floor and cleaning surfaces  Going shopping to refill fridge and buy more toilet paper  We should prepare for the lockdown and buy more toilet paper  Houses status:  Bedroom:  Window: Clean  Floor: Clean  Bed: Made  Surfaces: Clean  Kitchen:  Window: Clean  Floor: Clean  Fridge: Filled  Surfaces: Clean  Bathroom:  Surfaces: Clean  Toilet paper: More than enough | Houses status:  Bedroom:  Window: Dirty  Floor: Dirty  Bed: Unmade  Surfaces: Dusty  Kitchen:  Window: Dirty  Floor: Dirty  Fridge: Empty  Surfaces: Dusty  Bathroom:  Surfaces: Dusty  Toilet paper: Running out  Washing the windows and making the bed  Vacuuming the floor and cleaning surfaces  Going shopping to refill fridge and buy more toilet paper  We should prepare for the lockdown and buy more toilet paper  Houses status:  Bedroom:  Window: Clean  Floor: Clean  Bed: Made  Surfaces: Clean  Kitchen:  Window: Clean  Floor: Clean  Fridge: Filled  Surfaces: Clean  Bathroom:  Surfaces: Clean  Toilet paper: More than enough |
| **2** | <Run Program>  <User Inputs> | Houses status:  Bedroom:  Window: Dirty  Floor: Dirty  Bed: Unmade  Surfaces: Dusty  Kitchen:  Window: Dirty  Floor: Dirty  Fridge: Empty  Surfaces: Dusty  Bathroom:  Surfaces: Dusty  Toilet paper: Running out  Would you like to clean the house?  If yes then type 1 if not then type 2: 1  Alright let's get cleaning  Would you like to wash windows and make bed?  If yes then type 1 if not then type 2: 2  Alright not doing this task then  Would you like to vacuum floors and clean surfaces?  If yes then type 1 if not then type 2: 1  Vacuuming the floor and cleaning surfaces  Would you like to go grocery shopping?  If yes then type 1 if not then type 2: 5  Wrong input! Type 1 for Yes and 2 for No  Would you like to go grocery shopping?  If yes then type 1 if not then type 2: 1  Going shopping to refill fridge and buy more toilet paper  Would you like to hoard toilet paper?  If yes then type 1 if not then type 2: 2  Alright not doing this task then  Houses status:  Bedroom:  Window: Dirty  Floor: Clean  Bed: Unmade  Surfaces: Clean  Kitchen:  Window: Dirty  Floor: Clean  Fridge: Filled  Surfaces: Clean  Bathroom:  Surfaces: Clean  Toilet paper: Enough | Houses status:  Bedroom:  Window: Dirty  Floor: Dirty  Bed: Unmade  Surfaces: Dusty  Kitchen:  Window: Dirty  Floor: Dirty  Fridge: Empty  Surfaces: Dusty  Bathroom:  Surfaces: Dusty  Toilet paper: Running out  Would you like to clean the house?  If yes then type 1 if not then type 2: 1  Alright let's get cleaning  Would you like to wash windows and make bed?  If yes then type 1 if not then type 2: 2  Alright not doing this task then  Would you like to vacuum floors and clean surfaces?  If yes then type 1 if not then type 2: 1  Vacuuming the floor and cleaning surfaces  Would you like to go grocery shopping?  If yes then type 1 if not then type 2: 5  Wrong input! Type 1 for Yes and 2 for No  Would you like to go grocery shopping?  If yes then type 1 if not then type 2: 1  Going shopping to refill fridge and buy more toilet paper  Would you like to hoard toilet paper?  If yes then type 1 if not then type 2: 2  Alright not doing this task then  Houses status:  Bedroom:  Window: Dirty  Floor: Clean  Bed: Unmade  Surfaces: Clean  Kitchen:  Window: Dirty  Floor: Clean  Fridge: Filled  Surfaces: Clean  Bathroom:  Surfaces: Clean  Toilet paper: Enough |
| **4** | <Run Program>  <User Inputs> | Choose flavor Vanilla, Chocolate, Toffe: Toffe  All cookies baked  All cookies frosted  Flavor is wrong, everything has to be done again  All cookies baked  All cookies frosted  Flavor is wrong, everything has to be done again  All cookies baked  All cookies frosted  All cookies are correctly done  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe | Choose flavor Vanilla, Chocolate, Toffe: Toffe  All cookies baked  All cookies frosted  Flavor is wrong, everything has to be done again  All cookies baked  All cookies frosted  Flavor is wrong, everything has to be done again  All cookies baked  All cookies frosted  All cookies are correctly done  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe  Cookie is baked: True  Cookie is frosted: Toffe |
| **9** | <Run Program>  <User Inputs> | Zero div error  The argument does not contain numbers  Variable is not defined  Sorry, the file does not exist  Try using KeyboardInterrupt: KeyboardInterrupt exception is caught | Zero div error  The argument does not contain numbers  Variable is not defined  Sorry, the file does not exist  Try using KeyboardInterrupt: KeyboardInterrupt exception is caught |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Chart

Description automatically generated



Diagram

Description automatically generated

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.

[www.softwareideas.net](http://www.softwareideas.net)

[www.visual-paradigm.com](http://www.visual-paradigm.com)

[www.circle.visual-paradigm.com](http://www.circle.visual-paradigm.com)

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).

Pylint

Epydoc

Pylint png image:

Diagram

Description automatically generated

# Code

# File name: house.py

# Author: Steve Hommy

# Description: Create a House Class

class House:

    def \_\_init\_\_(self):

        self.bedroom\_window = "Dirty"

        self.kitchen\_window = "Dirty"

        self.bedroom\_floor = "Dirty"

        self.kitchen\_floor = "Dirty"

        self.bedroom\_bed = "Unmade"

        self.bedroom\_surfaces = "Dusty"

        self.kitchen\_surfaces = "Dusty"

        self.bathroom\_surfaces = "Dusty"

        self.kitchen\_fridge = "Empty"

        self.bathroom\_toilet\_paper = "Running out"

    def set\_wash\_windows\_and\_bed\_is\_made(self):

        self.bedroom\_window = "Clean"

        self.kitchen\_window = "Clean"

        self.bedroom\_bed = "Made"

    def get\_wash\_windows\_and\_bed\_is\_made(self):

        print("Washing the windows and making the bed")

        return self.bedroom\_window, self.kitchen\_window, self.bedroom\_bed

    def set\_vacuum\_floors\_and\_clean\_surfaces(self):

        self.bedroom\_floor = "Clean"

        self.kitchen\_floor = "Clean"

        self.bedroom\_surfaces = "Clean"

        self.kitchen\_surfaces = "Clean"

        self.bathroom\_surfaces = "Clean"

    def get\_vacuum\_floors\_and\_clean\_surfaces(self):

        print("Vacuuming the floor and cleaning surfaces")

        return self.bedroom\_floor, self.kitchen\_floor, self.bedroom\_surfaces, self.kitchen\_surfaces, self.bathroom\_surfaces

    def set\_shopping(self):

        self.kitchen\_fridge = "Filled"

        self.bathroom\_toilet\_paper = "Enough"

    def get\_shopping(self):

        print("Going shopping to refill fridge and buy more toilet paper")

        return self.kitchen\_fridge, self.bathroom\_toilet\_paper

    def set\_enough\_toilet\_paper(self):

        self.bathroom\_toilet\_paper = "More than enough"

    def get\_enough\_toilet\_paper(self):

        print("We should prepare for the lockdown and buy more toilet paper")

        return self.bathroom\_toilet\_paper

    def \_\_str\_\_(self):

        return f"""

    Houses status:

    Bedroom:

    Window: {self.bedroom\_window}

    Floor: {self.bedroom\_floor}

    Bed: {self.bedroom\_bed}

    Surfaces: {self.bedroom\_surfaces}

    Kitchen:

    Window: {self.kitchen\_window}

    Floor: {self.kitchen\_floor}

    Fridge: {self.kitchen\_fridge}

    Surfaces: {self.kitchen\_surfaces}

    Bathroom:

    Surfaces: {self.bathroom\_surfaces}

    Toilet paper: {self.bathroom\_toilet\_paper}

    """

# File: main.py

# Author: Steve Hommy

# Description: Main function

from house import House

def main():

    house = House()

    print(house)

    house.set\_wash\_windows\_and\_bed\_is\_made()

    house.get\_wash\_windows\_and\_bed\_is\_made()

    house.set\_vacuum\_floors\_and\_clean\_surfaces()

    house.get\_vacuum\_floors\_and\_clean\_surfaces()

    house.set\_shopping()

    house.get\_shopping()

    house.set\_enough\_toilet\_paper()

    house.get\_enough\_toilet\_paper()

    print(house)

main()

# File name: house.py

# Author: Steve Hommy

# Description: Create a House Class

class House:

    def \_\_init\_\_(self):

        self.bedroom\_window = "Dirty"

        self.kitchen\_window = "Dirty"

        self.bedroom\_floor = "Dirty"

        self.kitchen\_floor = "Dirty"

        self.bedroom\_bed = "Unmade"

        self.bedroom\_surfaces = "Dusty"

        self.kitchen\_surfaces = "Dusty"

        self.bathroom\_surfaces = "Dusty"

        self.kitchen\_fridge = "Empty"

        self.bathroom\_toilet\_paper = "Running out"

    def set\_wash\_windows\_and\_bed\_is\_made(self):

        self.bedroom\_window = "Clean"

        self.kitchen\_window = "Clean"

        self.bedroom\_bed = "Made"

    def get\_wash\_windows\_and\_bed\_is\_made(self):

        print("Washing the windows and making the bed")

        return self.bedroom\_window, self.kitchen\_window, self.bedroom\_bed

    def set\_vacuum\_floors\_and\_clean\_surfaces(self):

        self.bedroom\_floor = "Clean"

        self.kitchen\_floor = "Clean"

        self.bedroom\_surfaces = "Clean"

        self.kitchen\_surfaces = "Clean"

        self.bathroom\_surfaces = "Clean"

    def get\_vacuum\_floors\_and\_clean\_surfaces(self):

        print("Vacuuming the floor and cleaning surfaces")

        return self.bedroom\_floor, self.kitchen\_floor, self.bedroom\_surfaces, self.kitchen\_surfaces, self.bathroom\_surfaces

    def set\_shopping(self):

        self.kitchen\_fridge = "Filled"

        self.bathroom\_toilet\_paper = "Enough"

    def get\_shopping(self):

        print("Going shopping to refill fridge and buy more toilet paper")

        return self.kitchen\_fridge, self.bathroom\_toilet\_paper

    def set\_enough\_toilet\_paper(self):

        self.bathroom\_toilet\_paper = "More than enough"

    def get\_enough\_toilet\_paper(self):

        print("We should prepare for the lockdown and buy more toilet paper")

        return self.bathroom\_toilet\_paper

    def \_\_str\_\_(self):

        return f"""

    Houses status:

    Bedroom:

    Window: {self.bedroom\_window}

    Floor: {self.bedroom\_floor}

    Bed: {self.bedroom\_bed}

    Surfaces: {self.bedroom\_surfaces}

    Kitchen:

    Window: {self.kitchen\_window}

    Floor: {self.kitchen\_floor}

    Fridge: {self.kitchen\_fridge}

    Surfaces: {self.kitchen\_surfaces}

    Bathroom:

    Surfaces: {self.bathroom\_surfaces}

    Toilet paper: {self.bathroom\_toilet\_paper}

    """

# File: main.py

# Author: Steve Hommy

# Description: Main function

from house import House

def ask\_user():

    try:

        user\_input = int(input("If yes then type 1 if not then type 2: "))

        if user\_input == 1 or user\_input == 2:

            return user\_input

        raise ValueError()

    except ValueError:

        print("Wrong input! Type 1 for Yes and 2 for No")

def wash\_windows\_and\_bed\_is\_made(house):

    while True:

        print("Would you like to wash windows and make bed?")

        asking\_user = ask\_user()

        if asking\_user == 1:

            house.set\_wash\_windows\_and\_bed\_is\_made()

            house.get\_wash\_windows\_and\_bed\_is\_made()

            break

        elif asking\_user == 2:

            print("Alright not doing this task then")

            break

def vacuum\_floors\_and\_clean\_surfaces(house):

    while True:

        print("Would you like to vacuum floors and clean surfaces?")

        asking\_user = ask\_user()

        if asking\_user == 1:

            house.set\_vacuum\_floors\_and\_clean\_surfaces()

            house.get\_vacuum\_floors\_and\_clean\_surfaces()

            break

        elif asking\_user == 2:

            print("Alright not doing this task then")

            break

def shopping(house):

    while True:

        print("Would you like to go grocery shopping?")

        asking\_user = ask\_user()

        if asking\_user == 1:

            house.set\_shopping()

            house.get\_shopping()

            break

        elif asking\_user == 2:

            print("Alright not doing this task then")

            break

def enough\_toilet\_paper(house):

    while True:

        print("Would you like to hoard toilet paper?")

        asking\_user = ask\_user()

        if asking\_user == 1:

            house.set\_enough\_toilet\_paper()

            house.get\_enough\_toilet\_paper()

            break

        elif asking\_user == 2:

            print("Alright not doing this task then")

            break

def main():

    house = House()

    print(house)

    print("Would you like to clean the house?")

    while True:

        asking\_user = ask\_user()

        if asking\_user == 2:

            print("Alright I guess you are lazy for not cleaning the house")

            break

        elif asking\_user == 1:

            print("Alright let's get cleaning\n")

            wash\_windows\_and\_bed\_is\_made(house)

            vacuum\_floors\_and\_clean\_surfaces(house)

            shopping(house)

            enough\_toilet\_paper(house)

            break

    print(house)

main()

# File name: cookie.py

# Author: Steve Hommy

# Description: Create a Cookie Class

class Cookie:

    def \_\_init\_\_(self):

        self.bake = False

        self.frost = None

        self.flavor = ["Vanilla", "Chocolate", "Toffe"]

    def \_\_str\_\_(self):

        return f"""

        Cookie is baked: {self.bake}

        Cookie is frosted: {self.frost}

        """

    def set\_bake(self):

        self.bake = True

    def get\_bake(self):

        return self.bake

    def set\_frost(self, choice):

        self.frost = self.flavor[choice]

    def get\_frost(self):

        return self.frost

# File: main.py

# Author: Steve Hommy

# Description: Main function

from cookie import Cookie

import random

while True:

    flavor\_list = ["Vanilla", "Chocolate", "Toffe"]

    chosen\_flavor = input("Choose flavor Vanilla, Chocolate, Toffe: ")

    if chosen\_flavor in flavor\_list:

        break

def main(chosen\_flavor):

    choice = random.choice(flavor\_list)

    cookie\_list = []

    for object in range(10):

        object = Cookie()

        cookie\_list.append(object)

    # Bakes every cookie in the list one by one

    baked = 0

    for object in cookie\_list:

        object.set\_bake()

        baked += 1

        if baked == len(cookie\_list):

            print("All cookies baked")

            print()

    # Frosts every cookie in the list one by one

    frosted = 0

    for object in cookie\_list:

        object.set\_frost(choice)

        frosted += 1

        if frosted == len(cookie\_list):

            print("All cookies frosted")

            print()

    # Checks if the flavor is the same

    # If it's wrong do it all again

    test\_cookie = cookie\_list[0]

    if test\_cookie.get\_frost() != chosen\_flavor:

        print("Flavor is wrong, everything has to be done again")

        main(chosen\_flavor)

    # If flavor is right

    else:

        print("All cookies are correctly done")

        for object in cookie\_list:

            print(object)

main(chosen\_flavor)

# File: main.py

# Author: Steve Hommy

# Description: Main function

import random

filename = "Exercise8/exercise6/capitals.txt"

dictionary = {}

try:

    with open(filename) as file:

        for line in file:

            (key, value) = line.split()

            dictionary[key] = value

except FileNotFoundError:

    msg = "Sorry, the file " + filename + " does not exist.\n"

    print(msg)

try:

    while True:

        points = 0

        for i in range(10):

            country, capital = random.choice(list(dictionary.items()))

            print(country)

            answer = input("Give capital: ")

            if answer == dictionary[country]:

                print("Correct!\n")

                points += 1

            else:

                print("Wrong answer the correct answer is:", dictionary[country])

                print()

        print("Score:\n" + str(points) + "/10")

        break

except TypeError:

    print("Input must be a string")

# File: main.py

# Author: Steve Hommy

# Description: Main function

def zeroDiv(x, y):

    try:

        return x/y

    except ZeroDivisionError:

        print("Zero div error")

def valEr(var):

    try:

        return int(var)

    except ValueError:

        print("The argument does not contain numbers")

def naEr():

    try:

        print(x)

    except NameError:

        print("Variable is not defined")

def fiEr():

    try:

        with open("filename"):

            print("File found")

    except FileNotFoundError:

        print("Sorry, the file does not exist")

def keyInt():

    try:

        x = input("Try using KeyboardInterrupt: ")

    except KeyboardInterrupt:

        print ("KeyboardInterrupt exception is caught")

    else:

        print ("No exceptions are caught")

zeroDiv(1, 0)

valEr("fs")

naEr()

fiEr()

keyInt()