**CPSC 217: Assignment 3  ("The Dungeon of Doom": the Inner Sanctum)**

Due at 4 PM. For assignment due dates see the [main schedule](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/index.html#Main_grid:_lecture_&_tutorial_schedule,_assignment_information) on the course webpage. The program must be written and run under python version 3.X.

**New Concepts to be applied for the assignment**

* Decomposing a problem into functions (includes concepts such as parameter passing, return values, local variables/scope)
* If you want credit for your work: Do not use any pre-created functions/methods unless you are given explicit permission to do so. [[Generic list of allowable pre-created code in the form of python libraries](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/assignments/assignment3/index.html#Use_of_pre-created_Java_libraries0)]

Only new concepts that need to be applied in the assignment are listed, concepts previously applied in other assignments may need to used in the implementation of your solution.

**Assignment difficulty**

Students may find assignments more challenging than they first thought. It's best to start work as early as possible. Tips in the [very first lecture](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/notes/pdf/course_intro_admin.pdf)were provided but here's two reminders: 1) work through the lecture and tutorial material before looking in detail at the assignments 2) start work as soon as possible. If you find you cannot complete an assignment before the due date then you will not be granted an extension.

Note: it is not sufficient to just implement a working program and expect full credit. This is so you implement your solution in the correct way using good design principles and you apply the necessary concepts.**Even if your program is fully working and the program is not designed or implemented as specified in the assignment description (e.g. poor variable names used, named constants, functions not implemented appropriately or insufficiently etc.) then you will not be awarded full credit.**

Your program must be decomposed properly into functions. One way of decomposing your program is to implement the processing of instructions for a room as a single function or multiple functions. Because there are three rooms that will mean that **your program will consist of at least 4 functions** (1 per room plus a starting function). You may be able to subdivide your program using an alternate approach to writing 1 function per room but your program must follow [[principles of good design for using functions](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/assignments/assignment3/index.html#Function_specific_style_requirements_(principles_of_good_design_for_functions))].

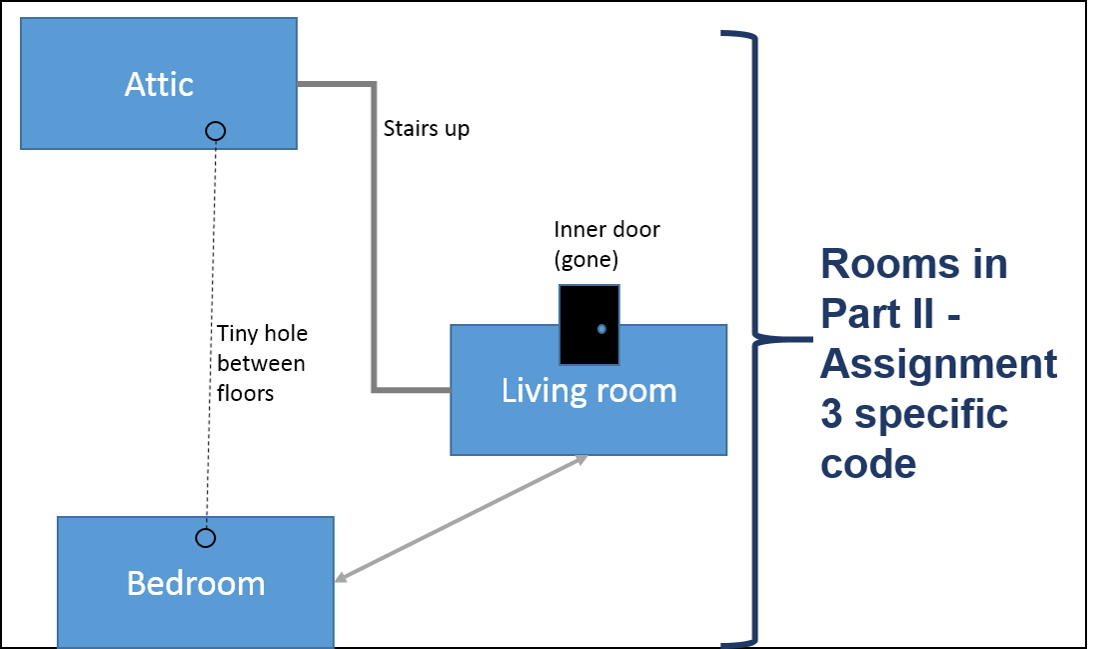
**Critical design requirements**

**All instructions must be enclosed within the body of a function, at least 4 functions must be properly defined. No global variables  may be employed.** The exceptions could include: import statements (not really needed for this assignment), the creation of global constants (e.g. ATTIC = 1), a global debugging flag and the call to the initial start or main function.

* **You will be penalized heavily if functions are not used or improperly used.**
  + No functions or a single function employed then the max assignment grade = 1.0 grade points
  + Two functions properly implemented then the max assignment grade = 2.0 grade points
  + Three functions properly implemented then the max assignment grade = 3.0 grade points
  + In order to qualify as properly implemented the function must include functionality that is appropriate to that function (e.g. anything related to the attic should not be implemented in a function that includes functionality for the bedroom) and not an empty (e.g. it just includes a pass instruction) or a token implementation (e.g. a few output statements or declarations).
* **In a similar fashion you will be penalized heavily if you define or use global variables**(global constants are okay, if you don't know the difference refer to the "Intro programming" lectures covered at the start of the term).
  + When **any** global variables (not a debugging flag and not constants) are employed then you will be penalized a full grade point (1.0). To rephrase: This penalty applies if you define one or more global variable(s).
  + Example:  
    location = "bedroom"  #Global declared  
    def processBedroom():  
        if (location == "bedroom"):  #Global referred to in the function.  
            #Etc.
* **What happens if you ignore both of these requirements (sufficient/proper use of functions and the prohibition on the use of global variables).**
  + **If you define no functions or just a single function and your program includes any global variables then you will be awarded no credit for this assignment**.
* **The function-related & global variable penalties are applied on top of other style penalties** but the lowest grade that may be awarded is a GPA of 0 (no negative scores will be awarded).

Contrary to some student rumors it is not the case that penalties exist in order to "curve the grades down" or to only allow a certain number of students to pass the course. Besides learning the mechanics of defining and calling functions you need to use them properly. For instances you shouldn't complete a course that teaches you how to write short stories and pen tales that are grammatically correct and may even use metaphorical references and include multiple characters if they include hackneyed metaphors and shallow character development. Because the penalties are quite strict then typically most/all (hopefully the latter) students will implement their solutions in the correct fashion.

**Functional requirements (for the marks allocated for each feature see the marking spreadsheet)**



**Living room contents:**a pot of soil, stairs going up, a dark entranceway, a ball of a string

* Viewing the pot of soil: 1) Default state when viewed, it looks dry  2) After it's been fertilized by the mouse (from the bedroom), when the player views the pot again a vine will grow that takes the player to paradise (game won and the program ends).
* Stairs going up: takes the player to the attic.
* Dark entranceway: takes the player into the bedroom.
* Ball of string: the player can pick up this object. There is only one ball of string, once it has been picked up the string should not appear again in this room.

**Attic contents:**a tiny hole in the floor, an unlimited supply of cheese, stairs going down.

* Hole: 1) player can pick up some cheese and try to drop it down the hole, the game indicates that the cheese is too big  2) If the player has the string then a new option is provided which is to drop the string down the hole. When this occurs the string should be removed from the player's inventory (the string doesn't reappear anywhere in the game, nor does the option to drop the string down the hole appear again in this room). The effect of dropping the string down the hole will be seen in bedroom.
* Cheese: player can pick some up, the supply of cheese is unlimited so repeatedly picking up the cheese won't diminish the supply (or change the room description). However the game should track that if the player is carrying any cheese (this will allow the player to feed the mouse this cheese in the bedroom).
* Stairs going down: takes the player back down to the living room.

**Bedroom contents:** a tomcat which is intently watching a mouse hole (default), mouse (after the cat leaves).

* Tomcat: when the player isn't carrying the string no interaction is displayed. When the player is carrying the string then there is a option to use the string to play with the cat. The result is the cat looks briefly at the player and then goes back to watching the mouse hole (the motion isn't enough to greatly distract the cat).
* Mouse: after the player drops the string down the hole from the attic the large motion distracts the cat sufficiently enough to leave the room and he takes the string with him. The mouse then comes out of its hole. Only if the player is carrying some cheese will the game display an option to feed the cheese to the mouse. When this occurs the mouse will leave the room and ahem *fertilize* the pot of soil. (The vine will grow and the player will win the game when the soil is viewed again from the living room). After fertilizing the soil the mouse will return to the room so there is the possibility of the player repeatedly feeding the mouse (the amount feed to the mouse is negligible so it doesn't change the amount carried by the player).  If the player hasn't picked up any cheese or if the mouse has been fed (without picking up any more cheese) then the room description should show that the mouse is in the room but there is no option to interact with it.
* Dark entranceway: takes the player into the living room.

As was the case with the previous assignment each room will provide a description of the contents, display a menu of options (which varies depending upon the actions of the player), get and error check the player's selection and display the menu and description for the room as long as the player remains in that location.

Game walkthrough: because this program is more complex than the previous one the summary map cannot provide details of what the player needs to do in order to win the game. Instead this step-by-step walkthrough specifies what's needed.

1. Living room: Pick up the ball of string.
2. Living room: go up the stairs to the attic.
3. Attic: drop the string down the hole.
4. Attic: pick up some cheese.
5. Attic: go down the stairs back to the living room.
6. Living room: go through the dark entranceway to the bedroom.
7. Bedroom: Feed the cheese to the mouse.
8. Bedroom: go through the dark entranceway back to the living room.
9. Living room: view the pot of soil (and win the game).

**Sample outputs of this program:**

You can find sample outputs of my solution to this assignment in a link to this [[folder](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/assignments/assignment3/outputs)].

In addition to grading on whether the above functionality was correctly implemented TAs will also look at documentation and style.

**Non-functional assignment requirements (style and documentation).**

* Python documentation: contact information (your name, student identification number and tutorial number). This should be specified in the header of the program (very top of your program in the form of Python documentation).
  + **Identifying information**: All assignments should include contact information (full name, student ID number and tutorial section) at the very top of your program.
  + **Program version** (dates are an acceptable alternative)
    - Under the version you should specify which assignment features were implemented in that version.
  + **Any program limitations or weaknesses**.
  + If you don't know how to document a program then refer to the "Intro to programming" section of the course.
  + **New documentation requirement starting with A3 and applies to all successive assignments**: [[Inline documentation](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/assignments/assignment3/inline_documentation.py)] (provided just before each function is defined): list the features of each room that were implemented in a particular function.
* Style requirements (-0.1 penalty for each now violation of a category there is no maximum cap on the number of style penalties that may be applied (5 categories of penalties means that a maximum penalty of -0.5 may be applied)
  + Naming conventions: You should employ good naming conventions for identifiers (variables, constants, function names, program file names).
  + **Named constants** should be used as appropriate (i.e. if they aren't defined and used when they should be then a penalty will be incurred).

|  |  |
| --- | --- |
| **Yes do it this way!** | **No. Not this way!** |
| #Defining 3 named constants **LEFT** = 0 **RIGHT** = 1 **CENTER** = 2 #Using the named constant 'RIGHT' if (silverLockPosition == **RIGHT**): | if (silverLockPosition == **0**): #What does 0 stand for??? |

* + The program code should have appropriate white space (specified in the "Intro to programming lecture') and alignment (specified throughout the lecture notes).
  + Code is self documenting e.g. Clear expressions (e.g. mathematical, Boolean). Good variable names and the use of named constants are examples of writing self documenting code but specifying clear expressions is important enough to be included in a separate category.
  + Of course if a student implements an extreme case of inefficient code (e.g. multiple loops or branches are used when one will do) then penalties may be applied but this is generally rare.
  + Your program should follow the 5 rules of thumb for designing user friendly software (distilled from Jakob Nielsen's 10 usability heuristics) which were included at the notes on 'Repetition' e.g. good error handling (such as prompts to the user to enter the required information clearly indicate what is required, good error messages should be provided for erroneous input) minimizing the user's memory load, being consistent, providing clearly marked exits & providing feedback as appropriate.
  + Having at least one violation in one of the above general style requirements will result in -0.1 penalty to marking. Multiple violations **in one category** still results in a single penalty (e.g. 3 bad variable names will still result in a -0.1 penalty). However violations **between categories** will result in cumulative penalties (e.g. a program that includes poor variable names, messy program layout and poor error handling will receive a -0.3 penalty)
* Function specific style requirements (principles of good design for functions) -0.2 penalty will be applied for each of the 3 function specific style requirements that have been violated.
  + Functions are one screen in length (normal screen resolution say ~30 lines max of code (excludes whitespace and documentation).
  + Functions implement one well defined task (e.g. processLivingRoomCommands() vs. processlivingRoomRunIntroductionRunConclusion()).
  + Code in one function is not duplicated in another function (not in the notes but this is just common sense that you don't write two functions where there's overlapping code - the overlap should likely be taken out of both functions and moved to another separate function). In this assignment there may appear to be some overlap (e.g. each room displays a menu of options but the specific options displayed will not be identical.
  + Violating any of the function specific style requirements will result in a -0.2 penalty for a violation in each of the 3 categories. E.g. a program that includes a function that exceeds the maximum length and implements two or more tasks would incur a -0.4 penalty.

**Reminder of critical style requirements**

**Don't forget that [**[**critical design requirements**](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/assignments/assignment3/index.html#Critical_design_requirements)**] (proper use of functions, no global variables) must also be followed.**

**Marking and grading**

* **What to submit**: In a typical semester submitted programs must run on the computer science network running Python 3.x. If you write you code in the lab and work remotely using a remote login program such as Putty or SSH then you should be okay (assuming you don't login to a non-Linux computer). If you choose to install Python on your own computer then it is your responsibility to ensure that your program will run properly here. **Remote learning version: If it won't run using Python 3.x on any computer then it won't be awarded credit**. It's up to you if you wish use the graphical program builder IDLE (or another development environment rather than a simple text editor) to write/run your programs but if you do you submit your program in the form of text ".py" file or files.
* **Late assignments or components of assignments:** will not be accepted for marking without approval for an extension beforehand. Alternate submission mechanisms (non exhaustive list of examples: email, uploads to cloud-based systems such as Google drive, time-stamps, TA memories) cannot be used as alternatives if you have forgotten to submit work or otherwise have not properly submitted into D2L. **Only files submitted into D2L by the due date** i**s what will be marked**, everything else will be awarded no credit. The final cut off date after which full assignments will not be accepted is after the [maximum progressive penalty (listed below)](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/assignments/assignment3/index.html#Late_submissions_(no_extension_granted):) can be applied. The final cut off date for mini-assignments is the due date because no late submissions are allowed for these smaller programs.
* **Extensions** may be granted for reasonable cases by the course instructor with the receipt of the appropriate documentation (e.g., a sworn declaration with a commissioner of oaths). Typical examples of reasonable cases for an extension include: illness or a death in the family. Example cases where extensions will not be granted include situations that are typical of student life: having multiple due dates, work commitments etc. Tutorial instructors (TAs) will not be able to provide extension on their own and must receive permission from the course instructor first. If you request an extension from me let me know the name of your tutorial instructor and the tutorial number because the markers won't accept late submissions without directly getting an email from me.
* **How you will be graded for full assignments**. As mentioned, as well as being marked on whether "your program works" you will also be marked on non-functional requirements such as style and documentation. Consequently this assignment will include a separate [[marking checklist](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/assignments/assignment3/A3_grading.xlsx)]. Besides seeing your grade point in D2L you can also see the detailed feedback that your TA will enter for each student. You can access the grading sheet in D2L under Assessments->Dropbox and then clicking on the appropriate assignment link. If you still cannot find the grading sheet then here is a [[help link](https://pages.cpsc.ucalgary.ca/~tamj/resources/markingFeedbackD2L/)]
* **Questions about grading afterward**: Assignments will be marked by your tutorial instructor (the "Teaching Assistant" or "TA") for your [[tutorial section](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/index.html#Tutorials_(teaching_and_help_sessions))]. When you have questions about marking this is the first person that you should be directing your questions towards. If you still have question after you have talked to your TA, then you can talk to your [[course (lecture) instructor](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/index.html#Instructor_contact_Information)] but please indicate in your email that you first contacted your TA.

**Method of submission:**

You are to submit your assignment using D2L [[help link](https://d2l.ucalgary.ca/shared/LE_Basics_Videos/index.htm)]. Make sure that you [[check the contents of your submitted files](https://pages.cpsc.ucalgary.ca/~tamj/resources/Verifying_D2L_Submissions.pdf)] (e.g., is the file okay or was it corrupted, is it the correct version etc.). It's your responsibility to do this! (Make sure that you submit your assignment with enough time before it comes due for you to do a check). If don't check and there were problems with the submission then you should not expect that you can "learn your lesson" and simply resubmit.

**D2L configuration for this course**

* Multiple submissions are allowed for this assignment (all versions are kept in D2L): You can (and really should) submit work as many times as you wish before the due date. However only the latest version of all the files is what will be marked, everything else will be ignored (because it is not fair to your marker to sort through multiple versions of your files).
* Do not use compression utilities (such as zip) or archiving utilities (such as tar) otherwise your submission may not be marked. The space savings in D2L is not worth the extra time required by the marker to process each submission.

**Late submissions for full assignments when there is no extension granted**: Make sure you give yourself enough time to complete the submission process so you don't get cut off by D2L's deadline (or your submission will be automatically flagged as late by D2L and it will be graded appropriately)..

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Submission received: | On time | Hours late : >0 and <=24 | Hours late: >24 and <=48 | Hours late: >48 and <=72 | Hours late: >72 and <=96 | Hours late: >96 |
| Penalty: | None | -1 GPA | -2 GPA | -3 GPA | -4 GPA | No credit (not accepted) |

**Collaboration:**

Assignments must reflect individual work; group work is not allowed in this class nor can you copy the work of others. Some "do nots" for your solution: don't publically post it, don't email it out, don't show it to other students.  For more detailed information as to what constitutes academic misconduct (i.e., cheating) for this course please read the following [[link](https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/assignments/misconduct.html)].

**Use of pre-created Python libraries:**

Unless otherwise told you are to write the code yourself and not use any pre-created functions (or methods). For most assignments the usual acceptable functions include: print(), input() and the 'conversion' functions such as int(), float(), str(). Look at the particular assignment description for a list of other functions/methods that you are allowed to use and still get credit in an assignment submission. If it's not listed then you should assume that you won't be able use the function and still be awarded credit. Note: This is a prohibition on using functions that someone else has wrote. You can (actually must) define your own functions for this assignment.