ITP-499: Python for Programmers

# Escape from SAL!

* Proposal - 2.0% of course grade
* Implementation – 18.0% of course grade

## Goals

* Learn the essentials of Tkinter and how its root function works
* Learn through documentation how to use a GUI
* Using conditional and for loop knowledge to create an organized flow of code structure (handlers, event loops)
* See the practical implications of File I/O
* Using data structure knowledge to create an organized system of data that can update through different events

## Overall Description

* The quarantine’s been over for a year now. We’re all back in person. Although isolation days seem long ago, what doesn’t seem like a long time ago? Spending all of your days in SAL. Recently, you have discovered something earth-shattering about the CS Department. What is it you ask? You would never reveal such a secret, but it is one so dastardly it would guarantee free tuition. Mark Redekopp knows that YOU know, but you only realize this after you’ve now been locked in SAL.

By avoiding eager CP’s who are trying to help you with any “questions” (or through attacking them if they become too annoying), giving chocolate from the vending machine to a janitor who just wants to be your friend, and gaining the key to the exit door, you just may make it out alive. You do have a time limit to escape and only so many health points the CPs can take by touching your character with their “questions” (their position intersects with yours).

Through using Tkinter, a splash screen, event loop logic/game handlers, data structures to store enemy/item information, and File I/O you will create the SAL experience yourself.

## Requirements

* Tkinter is used to support the game. A splash screen is present
* Game handlers with a main game loop are used for organization
* File I/O is utilized to help generate the world, as well as is written to for keeping track of the current high score
* Lists and/or dictionaries are used in the game’s enemy and item data structures
* There is both a win and a lose screen, and both take the player back to the splash screen.

## Sample Output

A piece of paper with writing on it

Description automatically generated

## Deliverables and Submission Instructions

* Modify this section as needed.
* Create a folder on your computer called **ITP499\_*FinalProject\_*LastName\_FirstName** (replace *LastName* with your last/family name and *FirstName* with your first name).
* Inside the folder, put your Python source code.
* Compress the folder (make a zip file). This cannot be done within PyCharm. Find the folder on your computer and compress it.
  + Windows:
    - Select your file
    - Right click
    - Send to ->
    - Compressed (zipped) folder
  + macOS:
    - Select your file
    - Right click
    - Compress
* Upload the zip file to your Blackboard section:
  + On Blackboard, click on the Assignments item in the course menu on the left.
  + Click on the specific item for this assignment.
  + Click on the Browse My Computer button and select your zip file.
  + Click the Submit button.

## Grading Criteria

* 0.5% Bulleted list with percentage followed by criteria. The total must add to 18.0%
* (4%) File I/O of reading in the current file works
  + (1%) Room aspects can spawn from the file
  + (.25%)Enemy’s proper attributes (movement bounds) correctly generate
  + (1%) Items on the ground are able to spawn
  + (.5%)“Goal” Door spawns where it should
  + (1.25%) All time high score is read off the file to show on the splash screen, and is updated as need be at the end of the game
* (8%) Game can create via Root Function and properly run the game loop
  + (.5%) The game window is able to draw itself
  + (.5%) There is an initial splash screen showing the high score, “start” and “help” buttons
  + (.5%) Depending if start or help is pressed, or if the player “dies” game state conditionals properly work
  + (1.5%) Work is subdivided in a logical fashion in the game, in proper event handlers (taking user input, updating the world, and then outputting)
  + (3%) Within the different handlers, data structures are consistently maintained to show new movement updates
  + (1%) If a player wins, a win screen is shown, and the player can press a button to return to the splash screen
  + (1%) If the player loses, a lose screen is shown with a button to go back to the splash screen
* (6%) Enemies can be attacked, items can be picked up (internal data is updated)
  + (1%) If an enemy dies, it is properly updated in the corresponding enemy data structure, and disappears from the world
  + (1%) If an item is picked up, it is removed from the data structure holding the world’s items, making it disappear off the screen and is placed into the player’s inventory.
  + (.5%) If an enemy is attacked, its health decreasing in the structure it is contained in
  + (.5%) If a player is attacked, its health decreases
  + (.5%) If a player gets additional health, this is also updated in the data structure
  + (2.5%) Enemies can move in different directions.