**Python Journal Template**

**Directions:** Follow the directions for each part of the journal template. Include in your response all the elements listed under the Requirements section. Prompts in the Inspiration section are not required; however, they may help you to fully think through your response.

Remember to review the Touchstone page for entry requirements, examples, and grading specifics.

**Name: Jordan**

**Date: May 7th, 2024**

**Final Replit Program Share Link:** [**https://replit.com/@grimmal72/Golf-Island#main.py**](https://replit.com/@grimmal72/Golf-Island#main.py)

***or editable link***::**https://replit.com/join/venkekkpte-grimmal72**

Complete the following template. Fill out all entries using complete sentences. Note, I followed this, but deviated a bit in the pseudocode part, to make it read more like code.

PART 1: Defining Your Problem

|  |
| --- |
| **Task**  State the problem you are planning to solve.  **Requirements**   * Describe the problem you are trying to solve for. * Describe any input data you expect to use. * Describe what the program will do to solve the problem. * Describe any outputs or results the program will provide.   **Inspiration**  When writing your entry below ask yourself the following questions:   * Why do you want to solve this particular problem? * What source(s) of data do you believe you will need? Will the user need to supply that data, or will you get it from an external file or another source? * Will you need to interact with the user throughout the program? Will users continually need to enter data in and see something to continue? * What are your expected results or what will be the end product? What will you need to tell a user of your program when it is complete? |
| *April 22nd thoughts*: I want to build a game out of if and else conditionals, and the “random” module. I am considering a **golf game**.  The **golf game**: It will have to be on a 2d plane, where magnitude can go up or down and that’s it, like a scalar (eg. temperature, weight in science). What I mean is, I don’t know how to code a 3d golf course in Python. So, if I make a golf game, it will use the “random” module, which determines how hard the ball is hit. Maybe there can be a rare chance of missing the ball with your club, or missing the hole with the ball. But it’s going to have to be like a left to right golf game like Ninja Golf for the Atari 7800. And, no visuals, just a text-based game. The ball can be hit closer to the hole, and it can maybe go past the hole, but it can’t really go backwards. Whether you get a birdie or albatross or what is kind of just up to the math library to determine.  *April 30th thoughts*: Some of my favorite media have been multi-franchise crossover concepts, or just entertainment media with highly varied character rosters.   * Super Smash Bros. is a fighting game with Mario/Bowser/Luigi/Donkey Kong/Peach from Super Mario, Link from the Legend of Zelda, Samus from Metroid, Pikachu from Pokemon, and many other characters. It has consistently been one of Nintendo’s biggest-selling games for the past 20 years. * Poker Night At The Inventory (and likewise Poker Night 2) by Telltale Games is a poker game with a crossover of characters including Strong Bad, Sam & Max, the Heavy (from Team Fortress 2), Ash (from Evil Dead), etc * Wacky Races was a 1960s cartoon show that was basically Mario Kart or Death Race 2000 but with Hanna Barbera cartoon characters. It had Scooby Doo, Dick Dastardly, Peter Perfect, the Gruesome Twosome, Yogi Bear, Quick Draw McGraw, and other characters.   **I’d like to make my golf game a crossover. There is no mention in the touchstone guidelines about copyright violations. I am obviously not going to sell this meager text-based golf game. Fair Use laws allow parody of original copyrighted material without consequence.**  The characters I am considering are Scooby Doo, Spider Man, Dick Dastardly from Wacky Races, GLaDOS from Portal, and two characters from Looney Tunes. They will have varying swing abilities, chances of missing the ball, of abandoning the game, putting the ball into a lake, etc. They will have things that they say before the game and throughout the game, kind of like how in the Pokemon games, the opponent will say something before, in the middle of, and after the fight.  There will be multiple courses choosable, and they will be objects derived from the course class. Likewise, the multiple characters will be objects derived from the character class. This will include the things they say, their names, and their ability stats.  The user can pick their character, and also pick their two opponent characters, choosing each from the larger roster. The same character can’t be played by multiple players. By the way, human multiplayer would be an interesting concept, but I’m not going to include that option. By multiple players I’m referring to the user and their AI opponents.  I want the wind to be able to slightly affect the ball. I want the course to affect the chance of landing in the rough, or into the lake. These things are features that I may not end up adding to the game, but I like the idea.  I want the game to be a bit unpredictable and cartoonish.  So there are 3 balls in play at the start of the game. There will be a primary game loop which checks if the balls are still in play, and if some have reached the hole. When a ball has reached the hole, the number of shots used for that ball is calculated, and compared to par. However, if there are still balls in play, the loop continues tracking those. The AI plays for itself, whereas the player presses a button to proceed each time. Maybe I’ll add a feature where you have to input a number, and how close you are to the randomly generated number determines how hard the club is swinged.  This is a 2d concept, in a scalar direction. If, say, the player is starting at zero, and the hole is at 1000, but they’ve passed it by reaching the number 1167, the next swing will subtract from the current number. So a swing of 200 will bring them to 967. Then the next swing, if it’s 50, will add to the sum, bringing us to 1017. Then a swing of 20 will bring us to 997. Then a swing of 5 will bring us to 1002. Make sense? The “holedistance” variable is like an equator that our swings gravitate to.  Preferably, the distances are like zones, where the character hits less hard the closer they are to the hole. So from the tee they shoot with a driver (long distance club, 1-wood, max 900 feet), within 1000 feet they shoot with a fairway wood (3-wood, max 600 feet), within 500 feet they shoot with wedges (max 320 feet), with 100 feet they shoot with the putter (max 50 feet, and divides in 2 every time you get closer to the hole.) |

PART 2: Working Through Specific Examples

|  |
| --- |
| **Task**  Write down clear and specific steps to solve a simple version of your problem you identified in Part 1.  **Requirements**  Complete the three steps below **for at least two distinct examples/scenarios**.   * State any necessary input data for your simplified problem. * Write clear and specific steps in English (not Python) detailing what the program will do to solve the problem. * Describe the specific result of your example/scenario.   **Inspiration**  When writing your entry below ask yourself the following questions:   * Are there any steps that you don’t fully understand? These are places to spend more time working out the details. Consider adding additional smaller steps in these spots. * Remember that a computer program is very literal. Are there any steps that are unclear? Try giving the steps of your example/scenario to a friend or family member to read through and ask you questions about parts they don’t understand. Rewrite these parts as clearly as you can. * Are there interesting edge cases for your program? Try to start one of your examples/scenarios with input that matches this edge case. How does it change how your program might work? |
| 1: Picking your course and character, and instituting some unpredictability and variability between playthroughs:  This aspect is important to me. I don’t like boring things. And when I play DOS or Apple II text adventure games, if it’s boring, I turn it off. Anyways, if there’s some complexity, it’s better than nothing. It’s better to have multiple characters and multiple courses than just one unnamed character and one unnamed course.  1a. (Picking your course)  The course selection menu will give us the course name, the hole distance from the tee point, and a description. Using a number, we can choose which course we want. An input() function will get the inputted number from the user, and store it in a variable. This is within a try statement within a while loop. If they input something that doesn’t work, we will except the error to stop the program from crashing, and ask them to reenter their answer. We will then enter a conditional statement that determines if the given number can map to one of the numbers that corresponds to a course in the list of the courses, using len().  That will be within a function. The function will be called when we choose the course.  By the way, how we see the courses like this in the first place is that some formatted strings and a “for” loop write out the courses and their details from the object, with their corresponding numbers beside them, based on the index value.  Whatever course is chosen will be saved in a variable called something like “selectedCourse”. From here we can do things like keep referencing the {selectedCourse.name} or {selectedCourse.holedistance}.  1b. (Picking your character)  The character selection function can be written almost exactly the same way as the course selection function. However, since I want there to be multiple players, I think it will be called multiple times.  The characters can be listed out beside their corresponding numbers and be chosen with an input(), same as the courses. But I want there to be multiple players, so the selected user character will be placed into the selectedUserCharacter variable, the selected opponent 1 will be placed into selectedEnemyCharacter1, and selected opponent 2 will be placed into selectedEnemyCharacter2.  These will be stored objects, determined by the user’s inputs, and they will be stored in a list.  “*players = [selectedUserCharacter, selectedEnemyPlayer1, selectedEnemyCharacter2]*”  Some other fun things may be done with the random variable, such as a chance of Spider Man abandoning the game to go fight crime. This would look like:  *def gameLoop():*  *~*  *if selectedUserCharacter == spiderMan or selectedEnemyCharacter1 == spiderMan or selectedEnemyCharacter2 == spiderMan:*  *num == random.randrange(0, 100)*  *if num == 1*  *print(“Announcer: What’s that? Spider Man is abandoning the game! He’s swinging away on his web! I think he’s off to fight crime!”)*  *else:*  *continue*  *~*  *~*  2.The game loop.  I’m unsure of how this will go. The list of current players will be consulted a lot. If a player has gotten their ball in the hole, their number of shots is calculated, and their turn is skipped indefinitely from that point forward.  Every time it’s someone’s turn, there is a chance of their character saying a snippy quote. Then they take their shot. Their swingrange is modified based on what club they’re using. A driver will shoot further than a putter.  How this works will be kind of like in an RPG, where there are base character stats, and then equipment that can create modifiers on the character’s stats (agility, strength, luck, etc.)  Kind of like the players list, or the courses list, there will be an equipment list.  A character’s swing power will be the product of a multiplication between base stat and equipment stat.  Eg.  Tweety Bird’s base swing power x Equipment power = Product  or  40 x 3.5 = 140 (for driver club)  40 x 1.8 = 72 (for wedge)  40 x 1 = 40 (for putter)  That is for max power. The 140 or 72 or 40, for instance, would be placed as the 2nd parameter of the “random.randrange()” method., eg “random.randrange(0, 72)”  The equipped piece of equipment will be stored in a variable.  Within each course, I may have the zones for each club within a list within their own course object.  Something triggers a player’s turn to end, and move onto another character’s turn. It might be the breaking of a loop, as certain circumstances like landing in a lake might require a character to continue their turn.  As I’ve stated above, once all the characters have gotten their ball in the hole, we move to the ending sequence, including the ranking of who came first. |

PART 3: Generalizing Into Pseudocode

|  |
| --- |
| **Task**  Write out the general sequence your program will use, including all specific examples/scenarios you provided in Part 2.  **Requirements**   * Write pseudocode for the program in English but refer to Python program elements where they are appropriate. The pseudocode should represent the full functionality of the program, not just a simplified version. Pseudocode is broken down enough that the details of the program are no longer in any paragraph form. One statement per line is ideal.   **Help with writing pseudocode**   * Here are a few links that can help you write pseudocode with examples. Remember to check out part 3 of the Example Journal Template Submission if you have not already. Note: everyone will write pseudocode differently. There is no right or wrong way to write it other than to make sure you write it clearly and in as much detail as you can so that it should be easy to convert it to code later. * <https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/> * <https://www.wikihow.com/Write-Pseudocode>   **Inspiration**  When writing your entry below ask yourself the following questions:   * Do you see common program elements and patterns in your specific examples/scenarios in Part 2, like variables, conditionals, functions, loops, and classes? These should be part of your pseudocode for the general sequence as well. * Are there places where the steps for your examples/scenarios in Part 2 diverged? These may be places where errors may occur later in the project. Make note of them. * When you are finished with your pseudocode, does it make sense, even to a person that does not know Python? Aim for the clearest description of the steps, as this will make it easier to convert into program code later. |
| * 6 characters, 3 stages, stored as referenceable objects with pullable attributes. * Print “Golf Island” * Press “y” to start (uses while True loop to wait until input validation passes) * Select your stage (uses numbers to choose, number must fall within range of available stages, ie length of list, or either ValueError exception or conditional will be triggered, and the loop continued) * Select your character. Lists them out with formatted strings, and then you choose your character with a number in much the same way as the stage. * selectedUserCharacter, selectedEnemyCharacter1, and selectedEnemyCharacter2 are stored in a list called players, to be referenceable. Using a for loop, this can access their attributes, such as by using “player.name”. * Wind speed is determined with a random magnitude between 1 and 25. The cardinal direction (such as north, south) of the wind will be also determined by random chance and pulled from a list of strings. This string will be used in validation against the cardinal direction of the hole direction from the tee point. * The angle will be calculated from the wind cardinal direction and the shooting cardinal direction, in the 360 radial plane. The stage’s hole direction will be pulled from the object attribute. If it’s a matching direction, the swing will be boosted. If the wind is pushing back on the ball, it could majorly affect your swing power. * The character will have a “swingpower” attribute that will be referenced, and compounded by the wind multiplier variable. This will be based on angle. Negative wind direction will be added to the swing power, much like how positive wind direction would be. This is how we wind up with near zero or near double magnitude swing powers. * Zones will be created as variables, and placed in a list. Clubs will also be created as variables, and placed in a list. If zone == 1, club = driver, etc. The clubs are shot multipliers, so will be used in math like “swingpower = swingpower \* driver” * If random.randrange() = a certain number, the character will say something after their swing. This calls a function, and involves a formatted string that references a large string in the character object. * Game loop: Uses a while True loop to run shotCycle on repeat for “players” (the list of the three players). * If it turns True, and if all players have finished playing, exit the loop and trigger the ending. * In the meantime, shotCycle runs each time a player is up * If player is finished playing, skip them. * random.randrange() causes chance of landing in lake. This will take you back to your last saved position and waste a turn. “player.position” and “player.numberofshotstaken” will be utilized. * random.randrange() causes chance of landing in rough. We want it to affect you next turn, not the same turn, though. Since local scope of the function may be forgotten, “inrough” will be stored in the player object. * If “player.position” == “holedistance”, they got the ball in the hole. * If “player.position” > “holedistance”, they’ll have to turn around and start shooting back towards where we tee’d from. * If “player.position” < “holedistance”, we’re still not even to or past the hole yet, we’re still shooting forward. * Once ending triggered, “player.numberofshotstaken” is put through a sort function to list the numbers from lowest to highest. These are placed in an ordered list called “scores”. * Best player chosen from spot in list. If third score out of range, it might be that there was a tie, making only a 1st and 2nd score fit within the three spots. * If characterScore = par - 1 they got a Birdie, etc. Say what each player got with a formatted string. * If Spider-Man quits the game, an exception is passed (using a class), and dialogue is given. It’s ugly, but easier than programming in a massive function to contain the entire game so that it can be exited if Spider-Man leaves. |

PART 4: Testing Your Program

|  |
| --- |
| **Task**  While writing and testing your program code, describe your tests, record any errors, and state your approach to fixing the errors.  **Requirements**   * For at least one of your test cases, describe how your choices for the test helped you understand whether the program was running correctly or not.   For each error that occurs while writing and testing your code:   * Record the details of the error from Replit. A screenshot or copy-and-paste of the text into the journal entry is acceptable. * Describe what you attempted in order to fix the error. Clearly identify what approach was the one that worked.   **Inspiration**  When writing your entry below ask yourself the following questions:   * Have you tested edge cases and special cases for the inputs of your program code? Often these unexpected values can cause errors in the operation of your program. * Have you tested opportunities for user error? If a user is asked to provide an input, what happens when they give the wrong type of input, like a letter instead of a number, or vice versa? * Did the outcome look the way you expected? Was it formatted correctly? * Does your output align with the solution to the problem you coded for? |
| **Error #1:**  I often got Replit “code is unreachable” errors due to me not initially understanding loops well enough. Sometimes they were just infinite loops. Sometimes it was that I was using “return False” to try to escape a “while True” loop However, I think I mixed up my syntax. “return” is for functions, yes? So I shouldn’t have been trying to use it to break a loop. I should have used “break” instead.    Within the interpreter, if I made a mistake with the loop, it would lead to an infinite text chain, or a SyntaxError.    But, again, when I used “break” instead of “return False” to exit the loop, the SyntaxError went away.    This is what it was supposed to look like if it works:    **Error #2:**    I kept getting this weird issue with this line of code.  “thirdBestScore = sorted(set(player.numofshots for player in players))[2]”  Something like IndexError: Index out of range.  The thing is, this doesn’t involve some other variable list. It’s just the three characters’ “numofshots” attribute. The problem seemed to be that the game saw matching “numofshots” values as the same thing, so a third place would actually go in the second index ([1]).  I ended up having to rewrite the endingSequence() function such that it was made adamantly clear that if there was a two-way tie in first place, third place might not exist, and other logical leaps.    **Error #3:**  I was getting infinite games due to a rounding error. I had to implement a conditional that triggers if the shotCycle swings the same number three times in a row. The shotMagnitudeHistory list will be viewed by the conditional, and while I was using ceil() to get round swinging numbers, if shotMagnitudeHistory had three repeats, it would switch the rounding function to floor(). This makes sure we aren’t having some issue like getting infinite “2” swings or infinite “0” swings.  I think it may also have been because I was allowing a chance of a windSpeed of zero at times. Since windSpeed never changes in my game (unless you re-run the program), and since windSpeed is used in the swingMagnitudeMultiplier, you can get stuck swinging zeroes forever. So I was getting infinite loops of a character being 2 feet away from the hole, swinging and never getting it in.  **Error #4:**  While implementing the solution to error #3, I was getting an UnboundLocalError by creating the variables repeatCount and shotMagnitudeHistory, because I chose to define them outside of the shotCycle() function. I ended up having to put “global” in front of them to get them to stop throwing errors. |

PART 5: Commenting Your Program

|  |
| --- |
| **Task**  Submit your full program code, including thorough comments describing what each portion of the program should do when working correctly.  **Requirements**   * The purpose of the program and each of its parts should be clear to a reader that does not know the Python programming language.   **Inspiration**  When writing your entry, you are encouraged to consider the following:   * Is each section or sub-section of your code commented to describe what the code is doing? * Give your code with comments to a friend or family member to review. Add additional comments to spots that confuse them to make it clearer.   **This is an almost 800-line program. If you’re going to try to read it, it’s best to do so on the Replit.** |
| # type: ignore  import math  import random  import time  # These classes and objects will be heavily utilized throughout the game. The 3 courses will belong to the Course class, and the 6 characters will belong to the Character class. I’m utilizing strings, booleans, etc, some of which will be read and updated every shotCycle(). shotCycle() will be defined much further down.  class Course:  def \_\_init\_\_(self, name, holedistance, description, par,  holedirectionfromtee):  self.name = name  self.holedistance = holedistance  self.description = description  self.par = par  self.holedirectionfromtee = holedirectionfromtee  rollingHills = Course("Rolling Hills", 804, "A very hilly golf course.", 18,  "East")  seaweedCove = Course("Seaweed Cove", 1020, "Watery.", 21, "North")  hauntedMansion = Course(  "Haunted Mansion", 948,  "Whose idea was it to turn a kilometer long mansion into a golf course?!",  16, "South")  class Character:  def \_\_init\_\_(self, name, numofshots, swingstats, motive, franchise,  swingpower, position, fancyscoreword, startquote,  afterswingquote, end1stquote, end2ndquote, end3rdquote,  inrough):  self.name = name  self.numofshots = numofshots  self.swingstats = swingstats  self.motive = motive  self.franchise = franchise  self.swingpower = swingpower  self.position = position  self.fancyscoreword = fancyscoreword  self.startquote = startquote  self.afterswingquote = afterswingquote  self.end1stquote = end1stquote  self.end2ndquote = end2ndquote  self.end3rdquote = end3rdquote  self.inrough = inrough  dickDastardly = Character(  "Dick Dastardly", 0,  "Uses cannons, magnets, and other nefarious tricks to attempt victory.",  "To melt the gold cup and sell its precious metal.", "Wacky Races", 250, 0,  " ", [  "Dick Dastardly: Hahahaha! Maybe I can set a booby trap to funnel the ball underground towards the hole! Then I'll win for sure! Oop... Drats... the ref's watching.\n",  "Muttley: \*Evil laugh\* // Dick Dastardly: Haha, that's right, Muttley, if we win this tournament, we'll melt the cup down and be rich!\n",  "Dick Dastardly: Can I bring my car on the course? \*whispers\* Muttley, get me the car. I'll drive the ball and get a hole in one. Drive! Get it? Hahahahahahahaha.\n",  "Dick Dastardly: Muttley, my boy, fetch me my lucky putter. We're going to win this cup fair or foul!\n",  "Dick Dastardly: I'll show these amateurs how a true villain fairs in competition. Watch and learn, fools!\n"  ], [  "Dick Dastardly: Yow! My back! Drats... I can't swing! Argh... get me the cannon, Muttley! We'll shoot the ball from here!\n",  "Dick Dastardly: Hahahaha! See you dumb dumbs at the finish line!\n",  "Dick Dastardly: Hahaha! Little do they know my ball is magnetic, and I've installed a supermagnet in the hole!\n",  "Dick Dastardly: Muttley, get me my jet-powered club!\n"  ],  "Dick Dastardly: Hahaha! We did it Muttley! We beat these fools, the gold is ours! We'll be rich!\n",  "Dick Dastardly: Hmmm... I can still sell the silver cup for money... Hahahahaha!! Next year I'll win this cup for sure!",  "Dick Dastardly: Drats!", False)  tweetyBird = Character(  "Tweety Bird", 0, "A bit on the weaker side.",  "Wants to make Sylvester the Cat jealous.", "Looney Tunes", 240, 0, " ", [  "Tweety Bird: Did I just see a puddy tat?\n",  "Yosemite Sam: Haha, little bird, how do you reckon you're going to swing that club, being, what, 4 inches tall? \*BONK!\* // Tweety Bird: What were you saying?\n",  "Bugs Bunny: Ahhhh... Keep your chin up, doc. // Tweety Bird: \*Takes out his shiny new golf club\*\n",  "Tweety Bird: Oh my, what a wuvwy day for a game of gowf.\n"  ], [  "Tweety Bird: Ooh, is that a worm? \*gobble\* I should have brought birdseed.\n",  "Tweety Bird: I may be wittle, but I pack a bwig swing!\n",  "Tweety Bird: Dis game is fun! I wike chasing the baww!\n"  ], "Tweety Bird: I won! I won! I'm a wittle biwd but I won!\n",  "Sylvester the Cat: \*snicker \*snicker\* \*snicker\* \*snicker\* // Tweety Bird: Alwight, waugh it up, buster.\n",  "Tweety Bird: AHHHHHHH!! THAT PUDDY TAT'S AFTER ME!!\n", False)  scoobyDoo = Character(  "Scooby Doo", 0, "He's a dog. He has decent swinging strength for a dog.",  "Money for Scooby Snacks", "Scooby Doo by Hanna Barbera", 280, 0, " ", [  "Scooby Doo: Ruh roh Raggy! How many shots is ris course?\n",  "Scooby Doo: Reah Raggy, rou're right, ris course is rearry rough. But I'll try my best!\n",  "Scooby Doo: Roinks! Was that a ghost?\n",  "Scooby Doo: Ret's get ready to sink some raskets. Er, I mean... sring some shots!\n",  "Velma: Scoob! You can't eat the golf ball!\n"  ], [  "Shaggy: Zoinks! Watch out for that lake, Scoob!\n",  "Scooby Doo: Rooby-Dooby-Doo!!\n"  ], "Scooby Doo: Ranybody want a Rooby Snack?\n",  "Scooby Doo: Dooby Dooby Doo!", "Scooby Doo: Ruh roh...\n", False)  spiderMan = Character(  "Spider-Man", 0,  "Superhuman, but with a chance of abandoning the game to fight crime.",  "A break from heroics", "The Avengers", 300, 0, " ", [  "Spider-Man: My spider senses tell me there's a... bird with a golf club... behind me. Uh... Back off, little bird.\n",  "Spider-Man: I'll have you all stuck to your seats in awe. \*poses for the camera\*\n",  "Spider-Man: Looks like we're about to swing into action. Aunt May, Uncle Ben, wish me luck.\n",  "Spider-Man: Even superheroes need a day off sometimes.\n",  "Spider-Man: Welcome to the Spider-Man Invitational. I could just swing from some trees and dunk the ball, but I'll go easy on you guys.\n"  ], [  "Spider-Man: My spidey senses are tingling... Uh oh, where's that ball headed?\n",  "Spider-Man: Fore!!\n",  "Spider-Man: Maybe I should go easy on these guys. It's not good for my PR to stunt on civilians. Jay Jonah Jameson's gonna trash me in the headlines.\n"  ], "Spider-Man: Maybe now they should call me Golfer Man.\n",  "Spider-Man: It's your... friendly neighborhood second placer? No no no... The Daily Bugle's gonna trash me in the headlines.\n",  "Spider-Man: Hmmm... I guess I can't be the hero all the time.\n", False)  GLaDOS = Character(  "GLaDOS", 0,  "Able to calculate exactly how to get a hole in one, but lacks the swing power. About equal in swing strength to Spider Man's superhuman strength.",  "To watch mortals get frustrated when they lose at sports.", "Portal", 300,  0, " ", [  "GLaDOS: Don't worry if you miss your shot. It's not like anyone will remember your performance anyway.\n",  "GLaDOS: I hope you're enjoying this golf competition. It's not like you have much else to do with your meaningless existence.\n",  "GLaDOS: Oh, how delightful, a group of mortals wiling away their limited time on Earth. Try to actually hit the ball.\n"  ], [  "GLaDOS: It seems my capabilities extend far beyond the confines of the laboratory.\n",  "GLaDOS: Calculating trajectory. Executing swing sequence.\n"  ],  "GLaDOS: Congratulations, or should I say, commiserations, to my pitiful competitors. At least you got a trophy for participation.\n",  "GLaDOS: This outcome is... not what I anticipated. You have my deepest admiration, mortal.\n",  "GLaDOS: Remarkable. You've managed to defeat me at golf. It won't happen again.\n",  False)  yosemiteSam = Character(  "Yosemite Sam", 0,  "Shoots the ball with his handguns, which is actually pretty inaccurate, but sometimes he can shoot it in midair, too.",  "No stranger to a gold rush, Sam wants the gold from the cup to fashion himself some gold pistols too.",  "Looney Tunes", 250, 0, " ", [  "Yosemite Sam: I reckon this here green ain't big enough for the both of us!\n",  "Yosemite Sam: Golf club? Dagnabbit, I'll shoot my way to the cup!\n",  "Yosemite Sam: Saddle up! Yarharharharharhar!\n",  "Yosemite Sam: I'm 'bout to wrangle me a trophy today, I reckon! Yeehaw!\n"  ], [  "Yosemite Sam: Yeehaw! That ball's headin' for the hole like a bat outta Hell!\n",  "Yosemite Sam: Git along, little ball, git along!\n"  "Yosemite Sam: \*BANG\* \*BANG\* Har har, I don't need a club, this is fun!\n"  ], "Yosemite Sam: This ain't my first rodeo!\n",  "Yosemite Sam: Well, har har har. Silver, huh? I reckon I can still fashion some nice silver for my hunting rifle.\n",  "Yosemite Sam: What in tarnation?! Alright, whose Mr. Big Shot?! No funny business, ya hear?!\n",  False)  courses = [rollingHills, seaweedCove, hauntedMansion]  characters = [  dickDastardly, GLaDOS, scoobyDoo, spiderMan, tweetyBird, yosemiteSam  ]  print("GOLF ISLAND - A GAME BY JORDAN MALMGREN\n")  startTheGame = " "  while startTheGame != "y":  startTheGame = input("Enter 'y' to start the game.\n")  if startTheGame.lower() == "y":  break  else:  print("Incorrect input. Enter 'y' when you'd like to start the game.\n")  continue  #this line is supposed to be where the loop ends/repeats  # The game is of course started based on the input of the startTheGame variable.  # What it contains will lead to starting the game or not starting the game.  #After the game has been started:  print(  "It's a sunny day on Golf Island. Golf Island is a prestigious golf tournament that happens once a year on a series of islands in the middle of the Pacific Ocean.\nYou're vying for the gold cup, which is made of real gold! Pick your stage:\n"  )  def displayCourses():  for i in range(len(courses)):  print(  f"{i + 1}.\n{courses[i].name}.\n{str(courses[i].holedistance)} feet from start to finish.\n{courses[i].description}\n"  )  # That above loop just lists out the stages. It isn't how the numbers are determined, by the way. The stages' numbers were already determined by the items' positions in the "courses" list.  # The below loop is just about trying to get a number key input that corresponds to the available options.  def selectCourse():  while True:  userCourseChoice = input(  "Enter the number of the course you'd like to play.\n")  try:  choice = int(userCourseChoice)  if 1 <= choice <= len(courses):  #Then the inputted course will be chosen. Except, as it turns out, the Haunted Mansion is old and collapses when you try to pick it.  if choice == 3:  print(  "Uh oh... Breaking news. The haunted mansion golf course just collapsed in on itself. Guess we won't be able to play there.\n"  )  else:  return courses[choice - 1]  else:  print(  "Invalid choice. Please enter a number within the range of courses available, using a number key.\n"  )  continue  except ValueError:  print(  "Invalid input. Please enter a number corresponding to the available stages, using a number key.\n"  )  continue  displayCourses()  selectedCourse = selectCourse()  print(f"You picked {selectedCourse.name}!\n")  # These next two loops work the same way as the stage selection loops.  def displayCharacters():  for i in range(len(characters)):  print(  f"{i + 1}.\n{characters[i].name}.\nFranchise: {characters[i].franchise}.\nMotive: {characters[i].motive}\nSwing stats: {characters[i].swingstats}\n"  )  def selectCharacter():  while True:  userCharacterChoice = input(  "Enter the number of the character you'd like to choose:\n")  try:  choice = int(userCharacterChoice)  if 1 <= choice <= len(characters):  # Return the chosen character  return characters[choice - 1]  else:  print(  "Invalid choice. Please enter a number corresponding to the available characters, using a number key.\n"  )  continue  except ValueError:  print("Invalid input. Please enter a number, using a number key.\n")  continue  print(  "Pick your player! Choose who you'd like to play as. Use the number key.\n"  )  displayCharacters()  selectedUserCharacter = selectCharacter()  print(  f"You selected {selectedUserCharacter.name}! You'll play as {selectedUserCharacter.name}.\n"  )  print("Select who will be opponent 1.\n")  selectedEnemyCharacter1 = selectCharacter()  while True:  if (selectedUserCharacter == selectedEnemyCharacter1):  print(  "That character is already in use! You'll have to pick a different character\n"  )  selectedEnemyCharacter1 = selectCharacter()  continue  else:  break  print(  f"You selected {selectedEnemyCharacter1.name}! Opponent 1 will be {selectedEnemyCharacter1.name}! Now pick opponent 2!\n"  )  selectedEnemyCharacter2 = selectCharacter()  while True:  if (selectedEnemyCharacter2  == selectedEnemyCharacter1) or (selectedEnemyCharacter2  == selectedUserCharacter):  print(  "That character is already in use! You'll have to pick a different character\n"  )  selectedEnemyCharacter2 = selectCharacter()  continue  else:  break  print(  f"You selected {selectedEnemyCharacter2.name}! Opponent 2 will be {selectedEnemyCharacter2.name}!\n"  )  print(  "You take a look around the green. A crowd cheers on the bleachers. A blimp passes by, and camera drones buzz around, filming the event. It's certainly a major festival. You look over at your competition.\n"  )  print(  f"Your opponents {selectedEnemyCharacter1.name} and {selectedEnemyCharacter2.name} stand in front of the tee.\n"  )  def randomEnemyQuote1ForStart():  enemyQuote1 = random.choice(selectedEnemyCharacter1.startquote)  print(enemyQuote1)  def randomEnemyQuote2ForStart():  enemyQuote2 = random.choice(selectedEnemyCharacter2.startquote)  print(enemyQuote2)  randomEnemyQuote1ForStart()  randomEnemyQuote2ForStart()  #As you can imagine, these two quote functions pull from the two above lists of potential enemy character phrases.  players = [  selectedUserCharacter, selectedEnemyCharacter1, selectedEnemyCharacter2  ]  windSpeed = random.randrange(1, 25)  # According to my Googling, wind above 40mph is a gale, and 74mph is a hurricane, so I'm not going to go that high.  cardinalDirections\_List = [  "North", "North East", "East", "South East", "South", "South West", "West",  "North West"  ]  windDirection = random.choice(cardinalDirections\_List)  holeDirection = selectedCourse.holedirectionfromtee  # I may add in the direction that the hole points from the tee as a cardinal direction within the course objects, and then based on the wind speed it will affect the ball placement.  if windSpeed == 1:  print("There's very little wind! Fantastic weather for a game!\n")  else:  print(  f"The wind is moving at {windSpeed} miles per hour, {windDirection.lower()}. The hole is {selectedCourse.holedirectionfromtee.lower()} of the tee. \n"  )  print(  f"The hole is {str(selectedCourse.holedistance)} feet away. Par is {str(selectedCourse.par)}. You tee up.\n"  )  # Calculate the angle between two cardinal directions  def calculateAngle(direction1, direction2):  directions = cardinalDirections\_List  angle = (directions.index(direction1) - directions.index(direction2)) \* 45  if angle > 180:  angle -= 360  elif angle < -180:  angle += 360  return angle  # Calculates the angle between windDirection and holeDirection  angle = calculateAngle(windDirection, holeDirection)  # Determine swing magnitude adjustment based on angle  if angle == 0:  swingAdjustment = 1 # No adjustment if wind and hole direction are the same  elif angle == 180 or angle == -180:  swingAdjustment = -1 # Detract from swing if wind and hole direction are opposite  elif abs(angle) < 90:  swingAdjustment = 1 - abs(  angle) / 90 # Boost or detract slightly based on angle  else:  swingAdjustment = -(abs(angle) -  90) / 90 # Boost or detract slightly based on angle  # Apply wind speed to swing magnitude adjustment  # The randrange was 0 to 25, but we want swingAdjustment to be 0 to 1 and potentially negative.  swingAdjustment \*= windSpeed / 25  # How this works is a swingMagnitude multiplied by one is a normal swing. swingAdjustment can be negative, and can cause us to be blunting our swing with something like a 0.78 \* swingMagnitude that slightly dulls the swing. Alternatively, you can have like a 1.98 \* swingMagnitude that almost doubles the power of the swing. But it can go no lower than a swingMagnitudeMultiplier of 0 and no higher than a swingMagnitudeMultiplier of 2.  swingMagnitudeMultiplier = 1 + swingAdjustment  swingMagnitudeMultiplier = round(swingMagnitudeMultiplier \* 10000) / 10000  print(  f"Due to the wind, your swing multiplier is: {str(swingMagnitudeMultiplier)}x."  )  # I was getting numbers like 0.040000000000000036x in the above print statement which is why I instituted the rounding function above it.  def endingSequence(players):  scoreNames = [  "Hole-in-one!", "Albatross!", "Eagle!", "Birdie!", "Par!", "Bogey!",  "Double Bogey!", "Triple Bogey!",  "Their score was worse than Triple Bogey, though..."  ]  def golfScore(shots, par):  if shots == 1:  return scoreNames[0]  elif shots <= (par - 3) and shots > 1:  return scoreNames[1]  elif shots == par - 2:  return scoreNames[2]  elif shots == par - 1:  return scoreNames[3]  elif shots == par:  return scoreNames[4]  elif shots == par + 1:  return scoreNames[5]  elif shots == par + 2:  return scoreNames[6]  elif shots == par + 3:  return scoreNames[7]  else:  return scoreNames[8]  # Initialize an empty list to store the fancyScoreWord for each player  fancyScoreWords = []  # Get each person's fancy score word (Such as "Birdie")  for player in players:  player.fancyscoreword = golfScore(player.numofshots, selectedCourse.par)  fancyScoreWords.append(player.fancyscoreword)  # Sort the three players' scores from lowest to highest, which is incidentally (in golf) best to worst.  scores = sorted(set(player.numofshots for player in players))  bestScore = scores[0]  # Check for existence of second and third scores  if len(scores) > 1:  secondBestScore = scores[1]  else:  secondBestScore = None  if len(scores) > 2:  thirdBestScore = scores[2]  else:  thirdBestScore = None  for player in players:  if player.numofshots == bestScore:  bestScorePlayer = player  elif player.numofshots == secondBestScore:  secondBestScorePlayer = player  elif player.numofshots == thirdBestScore:  thirdBestScorePlayer = player  winningPlayers = [] # Empty list to store the winners  for player in players:  if player.numofshots == bestScore: # If (and only if) player is #1  winningPlayers.append(  player) # then add them to the first index of winningPlayers  if len(winningPlayers) == 1:  # Prints the champion declaration.  print("Congratulations to the winner!")  print(  f"Player {winningPlayers[0].name} is the champion with {bestScore} shots! {bestScorePlayer.fancyscoreword}"  )  # If people are in second and third places:  if secondBestScore is not None:  print(  f"In second place is {secondBestScorePlayer.name} with {secondBestScorePlayer.numofshots} shots. {secondBestScorePlayer.fancyscoreword}"  )  if thirdBestScore is not None:  print(  f"In third place is {thirdBestScorePlayer.name} with {thirdBestScorePlayer.numofshots} shots. {thirdBestScorePlayer.fancyscoreword}"  )  else: # Happens if there was some kind of tie.  print("It's a tie between:")  for p in winningPlayers:  print(f"Player {p.name} is tied for the win!")  # Print 3rd place if it isn't a three-way tie but a two-way tie.  if thirdBestScore is not None:  print(f"In third place is {thirdBestScore}!")  if bestScorePlayer.name == dickDastardly.name:  print(f"{dickDastardly.end1stquote}")  elif bestScorePlayer.name == GLaDOS.name:  print(f"{GLaDOS.end1stquote}")  elif bestScorePlayer.name == scoobyDoo.name:  print(f"{scoobyDoo.end1stquote}")  elif bestScorePlayer.name == spiderMan.name:  print(f"{spiderMan.end1stquote}")  elif bestScorePlayer.name == tweetyBird.name:  print(f"{tweetyBird.end1stquote}")  elif bestScorePlayer.name == yosemiteSam.name:  print(f"{yosemiteSam.end1stquote}")  if secondBestScorePlayer.name == dickDastardly.name:  print(f"{dickDastardly.end2ndquote}")  elif secondBestScorePlayer.name == GLaDOS.name:  print(f"{GLaDOS.end2ndquote}")  elif secondBestScorePlayer.name == scoobyDoo.name:  print(f"{scoobyDoo.end2ndquote}")  elif secondBestScorePlayer.name == spiderMan.name:  print(f"{spiderMan.end2ndquote}")  elif secondBestScorePlayer.name == tweetyBird.name:  print(f"{tweetyBird.end2ndquote}")  elif secondBestScorePlayer.name == yosemiteSam.name:  print(f"{yosemiteSam.end2ndquote}")  if thirdBestScorePlayer.name == dickDastardly.name:  print(f"{dickDastardly.end3rdquote}")  elif thirdBestScorePlayer.name == GLaDOS.name:  print(f"{GLaDOS.end3rdquote}")  elif thirdBestScorePlayer.name == scoobyDoo.name:  print(f"{scoobyDoo.end3rdquote}")  elif thirdBestScorePlayer.name == spiderMan.name:  print(f"{spiderMan.end3rdquote}")  elif thirdBestScorePlayer.name == tweetyBird.name:  print(f"{tweetyBird.end3rdquote}")  elif thirdBestScorePlayer.name == yosemiteSam.name:  print(f"{yosemiteSam.end3rdquote}")  else:  print("It's a tie between:")  for p in winningPlayers:  print(f"Player {p.name} is tied for the win!")  print(f"In third place is {thirdBestScore}!")  print(  "Announcer: The Golf Island tournament will be back next year, everybody! Stay tuned."  )  print(  f"The crowd cheers, and confetti pops above our winners. A plane spells out {bestScorePlayer.name}'s name in the sky. What a beautiful day."  )  zone1 = "zone1"  zone2 = "zone2"  zone3 = "zone3"  zone4 = "zone4"  zone5 = "zone5"  zone6 = "zone6"  zones = [zone1, zone2, zone3, zone4, zone5, zone6]  currentZone = zone1  driver = 1.3  threewood = 0.8  iron = 0.5  wedge = 0.25  putter = 0.1  putterwithin10ft = 0.01  clubs = [driver, threewood, iron, wedge, putter, putterwithin10ft]  currentClub = driver  # Since this line is never revisited, it isn't an issue that we are seting currentClub to driver, as it will just be reassigned later.  # I've been having an unfortunate problem where the necessary ceil function for rounding leads to infinite "2" shots. That is, swings with position changes of zero. If I use the floor function for rounding, I sometimes get infinite "0" shots. The solution? To implement the bottom three variables for usage in a conditional. If a shot magnitude is repeated three times, the character will switch to the opposite function, which will hopefully resolve the infinite rounding error.  repeatThreshold = 3  shotMagnitudeHistory = []  repeatCount = 0  def shotCycle(player, hole\_distance):  if player.position == hole\_distance:  # If the player has already reached the hole, ignore them  return  # The next three conditional blocks are used to check what zone we're in and what club we're using. The reason I get that sorted before taking a swing is so that we pick the right club for the current position BEFORE swinging each time, instead of after, which wouldn't make much sense.  if player.position >= 0 and player.position <= (hole\_distance \* 0.30):  currentZone = zone1  elif player.position > (hole\_distance \*  0.30) and player.position <= (hole\_distance \* 0.50):  currentZone = zone2  elif player.position > (hole\_distance \*  0.50) and player.position <= (hole\_distance \* 0.70):  currentZone = zone3  elif player.position > (hole\_distance \*  0.70) and player.position <= (hole\_distance \* 0.85):  currentZone = zone4  elif player.position > (hole\_distance -  10) and player.position < (hole\_distance + 10):  currentZone = zone6  else:  currentZone = zone5  if currentZone == zone1:  currentClub = driver  elif currentZone == zone2:  currentClub = threewood  elif currentZone == zone3:  currentClub = iron  elif currentZone == zone4:  currentClub = wedge  elif currentZone == zone5:  currentClub = putter  else:  currentClub = putterwithin10ft  swingPowerBasedOnClub = player.swingpower # Despite the finality that this line seems to have, the variable is about to have a different value.  # By the way, the reason I use this variable instead of player.swingpower is that with player.swingpower it was taking the assigned multiplied value into the next loop, after which it would multiply to be even bigger the next turn, growing or shrinking exponentially.  if currentClub == driver:  swingPowerBasedOnClub \*= driver  elif currentClub == threewood:  swingPowerBasedOnClub \*= threewood  elif currentClub == iron:  swingPowerBasedOnClub \*= iron  elif currentClub == wedge:  swingPowerBasedOnClub \*= wedge  elif currentClub == putter:  swingPowerBasedOnClub \*= putter  else:  swingPowerBasedOnClub \*= putterwithin10ft  # Player takes a shot  if player == selectedUserCharacter: # If it's the user's turn  while True: # Loop doesn't end until you properly input a number key  try:  swingMagnitude = " "  swingMagnitudeInput = int(  input(f"Enter your shot with the number key: "))  if 1 <= swingMagnitudeInput <= 25:  swingMagnitude = swingMagnitudeInput \* int(swingPowerBasedOnClub)  abs(swingMagnitude -  (random.randrange(1, 25) \* int(swingPowerBasedOnClub)))  break  else:  print("Select a number key from 1 to 25.")  continue  except ValueError:  print("Invalid input. Please enter a number using the number key.\n")  continue  else:  swingMagnitude = random.randrange(1, int(swingPowerBasedOnClub))  if player.inrough == True:  print(  f"{player.name}'s ball was in the rough, so this shot might not be ideal..."  )  swingMagnitude = int(swingMagnitude / 2)  # If the ball is in the rough, inRough will be set to true. This function isn't called until the after the ball has landed. However, if set to true, that player's property will be adjusted to player.inrough = true. swingMagnitude will be halved.  def isBallInRough():  probability = 1  if random.randrange(0, 25) < probability:  print(  "Uh oh! The ball fell in the rough! It's gonna be hard to get a good shot from there next turn!"  )  player.inrough = True  def randomVerb(  ): # This is for the sentence that gets used every time a player strikes the ball.  verbNum = random.randrange(1, 6)  if verbNum == 1:  verbChoice = "swatted"  elif verbNum == 2:  verbChoice = "swung"  elif verbNum == 3:  verbChoice = "struck"  elif verbNum == 4:  verbChoice = "hit"  elif verbNum == 5:  verbChoice = "thwacked"  elif verbNum == 6:  verbChoice = "knocked"  return verbChoice  verb = randomVerb()  # Update player's position based on the number received from the swingMagnitude randrange.  print(f"{player.name} just {verb} the ball {swingMagnitude} feet.")  # Pause to simulate turn  time.sleep(0) # Adjust the time as needed  # This is mostly just here from the debugging process. It can still be used to slow down how fast the text is printed by putting in a 1 or a 2..  # Save player position before the swing, so we can reference it if we sink the ball by accident  lastSavedPlayerPosition = player.position  # These throw an error if not accessible.  global repeatCount  global shotMagnitudeHistory  # Whether or not we're shooting towards the hole (adding to the sum) or shooting down towards the hole (subtracting from the total).  #swingMagnitudeMultiplier was defined based on the wind, that's it.  if shotMagnitudeHistory.count(swingMagnitude) >= repeatThreshold:  if (player.position < hole\_distance):  player.position += math.floor(swingMagnitude \* swingMagnitudeMultiplier)  else: # (player.position > hole\_distance)  player.position -= math.floor(swingMagnitude \* swingMagnitudeMultiplier)  repeatCount = 0 # Reset repeated count  shotMagnitudeHistory = []  else:  if (player.position < hole\_distance):  player.position += math.ceil(swingMagnitude \* swingMagnitudeMultiplier)  else: # (player.position > hole\_distance)  player.position -= math.ceil(swingMagnitude \* swingMagnitudeMultiplier)  repeatCount += 1 # Increment repeat count  # Add current shot magnitude to history  shotMagnitudeHistory.append(swingMagnitude)  # Each character's number of shots starts at 0, but after a shot, it goes up by one. If the ball goes in the lake, it still goes up by one. This is the only line of code we need to do this.  player.numofshots += 1  # Chance of ball falling in lake  def isBallInLake():  probability = 1  if random.randrange(0, 25) < probability:  print(  "Uh oh! The ball fell in a lake! They'll have to shoot from where they were last positioned."  )  player.position = lastSavedPlayerPosition  else:  return None  isBallInLake()  player.inrough = False  isBallInRough()  # Check if the player has reached the hole, or print current distance from hole, or print how much they overshot it by.  feetRemaining = hole\_distance - player.position  if feetRemaining > 0:  print(f"{player.name} is {feetRemaining} feet away from the hole.")  elif feetRemaining == 0:  print(f"{player.name} has reached the hole!\n")  else:  print(  f"{player.name}'s ball is {-feetRemaining} feet further than the hole! They'll need to backtrack a bit."  )  # This is the function that makes it so that the player sometimes speaks after hitting the ball.  def afterSwingSpeakChance(player):  probability = 1  if random.randrange(0, 10) < probability:  afterSwingQuote = random.choice(player.afterswingquote)  print(f"{afterSwingQuote}")  else:  return None  afterSwingSpeakChance(player)  # \*this is the end of shotCycle\*  def gameLoop(players, hole\_distance):  allBallsInHole = False  while not allBallsInHole:  class SpideyException(Exception):  pass  def spideyEnding():  print(  "Spider-Man: Uh oh... I'm getting a ring. \*BZZT\* \*BZZT\* Hello? Yes, Spider-Man speaking... There's a bank robbery? And it's the Green Goblin? I'll be there in a jiffy."  )  print(  "Announcer: This just in. Spider-Man is abandoning the game. He's off to fight crime. The game will have to be put on hold until he's back. Wait, he's leaving the island? Where's he going, New York? Oh my goodness. What, is he gonna swing across the ocean?"  )  raise SpideyException(  "\n \n SPIDER-MAN ABANDONED THE GAME TO GO FIGHT CRIME! \n J. Jonah Jameson: Darn that Spider-Man! He ruined the Golf Island Tournament! I'm gonna drag his name through the mud in the headlines tomorrow for sure!"  )  if spiderMan in players:  # Chance of ending the game due to Spider Man abandoning to go fight crime.  spideyProbability = 1  chanceOfHeroicDuties = random.randrange(0, 120)  if spideyProbability == chanceOfHeroicDuties:  spideyEnding()  for player in players:  shotCycle(player, hole\_distance)  # Check if all balls are in the hole  allBallsInHole = all(player.position == hole\_distance  for player in players)  # If all the characters' balls have reached the holes, the while not loop will be broken, and allBallsInHole will be permanently set to True, at least within the scope of the gameLoop function.  endingSequence(players)  # \*this is the end of gameLoop\*  gameLoop(players, selectedCourse.holedistance)  # Note: I'm happy with how this turned out. The only thing I want there to be is a restart game option at the ending screen, but it's fine. |

PART 6: Your Completed Program

|  |
| --- |
| **Task**  Provide the Replit link to your full program code.  **Requirements**   * The program must work correctly with all the comments included in the program.   **Inspiration**   * Check before submitting your touchstone that your final version of the program is running successfully. |
| <https://replit.com/join/venkekkpte-grimmal72> |