

Due Feb 15 by 11:59pm**Points** 100**Available** after Feb 6 at 10am**Home****Resources**

Homework 3 Booleans, Decision Making & Repetition (Loops)

This is an introductory course, and we want to make sure that everyone has a solid understanding of the fundamentals, so we sometimes rule some Python tools in or out. For this homework, you can only use while loops for iteration (no range loops or for loops this time). Also, data structures that we have not covered yet (e.g. Lists, Dictionaries, Tuples, etc.) are off-limits for this assignment. Do not use them.

For the code portion of this assignment, you are responsible for adhering to our [style guide](#) sections CS1-CS7.

Written Component (20% of your homework score)

- Filename: written.txt

Please open a plaintext file (you can do this in IDLE or any plaintext editor you like, such as TextEdit or NotePad) and type out your answers to the questions below. You can type your

answers into Python to confirm but answer the questions first!

Written #1

For each of the loops below, specify:

- Whether the loop is infinite
- Whether an error would result
- If neither of the above, what would be printed to the terminal? Be specific with spacing, indentation, and line breaks.

1A

1	<code>level = 0</code>
2	<code>danger = 3</code>
3	<code>while level < danger:</code>
4	<code> print("Danger, Will Robinson!")</code>
5	<code> level += 1</code>

1B

1	<code>level = 0</code>
2	<code>danger = 3</code>
3	<code>while level > danger:</code>
4	<code> print("Dr. Smith")</code>
5	<code> level += 1</code>
6	<code>print("Lost in Space!")</code>

1C

1	<code>level = 0</code>
---	------------------------

```
2  danger = 3
3  while level < danger:
4      print("Jupiter-2", end = " ")
5  print("Identify yourself, please")
```

Written #2

2A Which of the following two options will correctly prompt the user until they enter a valid weekend day?

Option 1

```
SAT = "Sat"
```

```
SUN = "Sun"
```

```
day = input("Which day is better for brunch?\n")
```

```
while day != SAT and day != SUN:
```

```
    day = input("Sorry, what? Please enter a valid day.\n")
```

```
print("Thank you!")
```

Option 2

```
SAT = "Sat"
```

```
SUN = "Sun"
```

```
day = input("Which day is better for brunch?\n")
```

```
while day != SAT or day != SUN:
```

```
    day = input("Sorry, what? Please enter a valid day.\n")
```

```
print("Thank you!")
```

2B For the option above that doesn't work (i.e., it doesn't successfully detect a valid weekend day), is the problem that...


1. It thinks bad inputs (like "Mon") are valid.
2. It thinks good inputs (like "Sat") are invalid.

Programming Component (80% of this HW)

Code Structure and Style

A percentage of your score for every homework is for writing good, clear, readable code. For HW3, focus on writing good loops and clear conditional statements. The only kind of loop allowed on this HW is the while loop.

Programming #1 (25%)

- Filename: `shape_clicker.py`
- Starter file: [PositionService.py](https://northeastern.instructure.com/courses/102990/files/13111278/download?download_frd=1) 
(https://northeastern.instructure.com/courses/102990/files/13111278/download?download_frd=1)
- Starter file: `shape_window.png`



This part of the homework builds on our shape drawing skills from last week and add some and event-driven programming to what we did. Don't worry, you do NOT have to reuse your code from last week unless you want to.

NO FLOWCHART or TEST CASES REQUIRED.

Your program should do the following:

1. Use the same background image as for Homework 2 – Align Draw. Remember, the .png image of the application window is 970x635 pixels.
Note: Most recent distributions of Python on most operating systems allow the use of .png for background images. However, a few students reported issues with their systems requiring .gif instead of .png for the image. For this assignment if you encounter difficulties with the .png, you are permitted to convert it to a .gif as long as you maintain the original image size.
2. Draw a circle with a RADIUS of 80 starting at (0, 0). Use green for your pen color.
3. Register for mouse clicks
4. When you click anywhere in the circle with your mouse, erase the circle
5. When you click anywhere on the canvas, draw the circle, with the center at the (x, y) coordinates where you clicked

Note: You may use the Turtle/pen as a global variable. However, other than any constants you use, no variables should be declared in the global namespace.

We have provided the `PositionService` component for your convenience. It acts as a "global ledger" for you to easily write and read position data so that you won't need to create global variables to handle that task.

Important: Use of the `PositionService` is optional! For this assignment you can "opt-in" and use the `PositionService`, or you can "do your own thing" and create your own approach to managing the data.

IF you decide to make use of the `PositionService`:

Import it into your program. Again, `PositionService` acts as a "position ledger" that allows you to store and retrieve information about your circle regarding its position and visibility.

`PositionService` is implemented using object-oriented programming, so you probably won't understand the code yet. Don't worry about that, simply use the service via these functions:

```
set_position_x( x )    # set the x position of your circle
set_position_y( y )    # set the y position of your circle
set_position(x, y)     # set both x and y positions of your circle
get_position_x()       # get the x position
get_position_y()       # get the y position
is_visible()           # returns True/False if you've "hidden" your shape
                        # by erasing it.
                        #YOU are responsible for setting the visibility
with set_visible()
set_visible( visibility ) # Pass in a Boolean to indicate that you've
hidden or displayed your shape
```

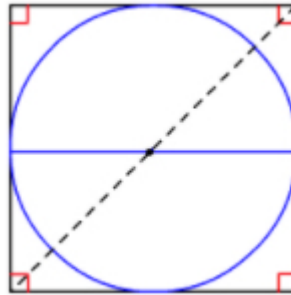
By convention, True is visible, False is

hidden

Keep in mind: The PositionService is like a ledger book you're writing in. It's not automatic. Every time you change the position of your circle, YOU are responsible for calling the appropriate `set_position()` function(s) to store the new (x, y) coordinates of the circle. Likewise, before you store the new coordinates, be sure to retrieve the current (x, y) with `get_position` if you need to, so you know what to erase from the canvas.

Other Notes:

- Remember our inscribed circle from last week. For our “click capture”, you are allowed to use the surrounding square as your “area for the circle”. In other words, if I click in the upper right corner of the square below, that would be an allowable “click capture” for the circle in this program despite the fact that it is properly outside of the circle’s region. For this assignment, it’s close enough.

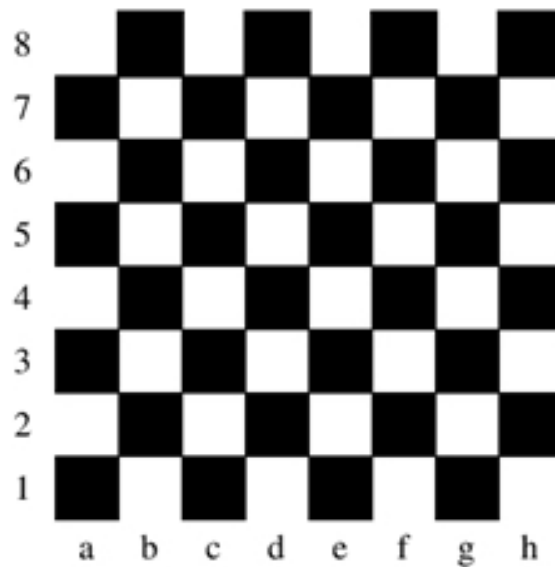


- It's very difficult to test graphics-related functions without explicit test tools (screen-grabbers, functional test tools, etc.), so your most important job is to ensure your turtle movement and drawing is as consistent as possible. Of note, when you move the turtle, it has a directional heading (N/S/E/W). Take care to set the heading consistently to ensure repeatable drawing of the circle
- (Debugging Hint): If you're having trouble getting the graphics coordinates correct, try printing each (x, y) clicked by the user. Using this approach may help you track down common errors like flipped x-y coordinates, wrong offsets, etc.

Programming #2 What Color is That Square? (25%)

- File names: `color_square.py`, `chessboard.py`, `test_squares.py`

Positions on a chess board are identified by a letter and a number. The letter identifies the column; the number identifies the row. See the image below.



You will provide the functions that allow us to determine:

1. if a chessboard square is either black or white, given its coordinates.
2. If the row/column is valid or not for the standard chessboard

For example, if we pass in "d" for the column and 4 for the row, your function that determines color should return the **string**: "**BLACK**". If we pass in "k" for the column to your function that validates columns, that function should return the **boolean** **False**.

Requirements:

- Your function for validating the column should be able to accept upper **or** lowercase strings as parameters.
- Your functions for validating the row should be able to accept integers **or** strings as parameters.
- Your function that determines black-or-white may assume that the input has been validated prior to us calling it (that is a *precondition*) and should return the string "BLACK" or "WHITE" based on the row/column values passed in
- **You must provide three "helper functions". Two of those functions:**
 - `check_valid_row(row)` and
 - `check_valid_column(column)`

must be defined in the file `chessboard.py`:

The **third** function: `black_or_white(row, column)` must be defined in the file `color_square.py`

- `check_valid_row` and `check_valid_column` both return a **boolean** – **True** if the row or column passed in is valid, **False** if not.

Remember to define these functions in a `chessboard.py`. We will run auto-tests on these functions, so be sure to name them properly and place them in the correct files.

- **You must write test function for `check_valid_row()` and `check_valid_column()`.** Your test functions should validate a few "normal" scenarios as well as one or two "out of bounds" situations for the rows and columns. You may name your test functions whatever you wish, but you must define them in a file named `test_squares.py`. In this same file, also include a function called `test_squares()` that acts as the "test driver" and calls your two test functions. Our auto-grader will use this function to interact with your test suite.

You have flexibility in how you present your test results, but conciseness and ease-of-readability (or lack thereof) will influence your grade. Here's an example of what you might consider producing:

```
Column: i, Expected: False, Actual: False
Column: A, Expected: True, Actual: True
Column: B, Expected: True, Actual: True
...
Row: 1, Expected: True, Actual: True
Row: 2, Expected: True, Actual: True
Row: 3, Expected: True, Actual: True
```

Your grade for this part of the assignment will be based on your solution code, your test functions provided and our own test suite, as well as manual inspection of your solution code for other rubric items (documentation, efficiency, etc.).

No `main()` is required for this assignment.

Resources: You might find the built-in `ord()` and `chr()` functions, as well as an ASCII table of characters helpful for this part of the assignment.

Look here: <http://www.asciitable.com/> [_ \(http://www.asciitable.com/\)](http://www.asciitable.com/)

Programming #3 Pokemon Tournament (30%)

- File names: `pokemon.py`

Pokemon was a worldwide cartoon, video game and card-game phenomenon. My nephews used to play the card game for hours every day. For this problem, you will write a program that runs a Pokemon tournament

Rules:

We'll use a variation of Rock-Paper-Scissors (RPS) to manage each Pokemon battle. The rules are simple and work as follows:

- Rock vs. Scissors: Rock wins.
- Paper vs. Rock: Paper wins.
- Scissors vs. Paper: Scissors wins.

At the beginning of each round, a player secretly makes their selection (rock, paper or scissors). Simultaneously the players display their choices. If both players display the same selection, it's a draw. Otherwise the winner is determined by the RPS "what beats what" designation above.

The names of the Pokemon for this tournament are as follows

- Bulbasaur, Charmander, Butterfree, Rattata, Weedle, Pikachu, Sandslash, Jigglypuff, Raichu, Diglett

You can find these and other characters from the Pokemon universe here:

<https://www.ranker.com/list/complete-list-of-all-pokemon-characters/video-game-info>
(<https://www.ranker.com/list/complete-list-of-all-pokemon-characters/video-game-info>)

Requirements:

The order of the pokemons listed above is important. Bulbasaur is pokemon 1, Charmander is pokemon 2 and so on. The special case is Diglett, who is the "default" pokemon.

For this game, use the following values: `Rock == 1`, `Paper == 2`, and `Scissors == 3`

- **You must provide two functions** as part of your solution:
 - `get_player(num)` and
 - `check_battle(computer, player)`

`get_player()` returns the name of the pokemon associated with the numbering scheme described above. For example, calling `get_player(2)` returns Charmander. Calling `get_player(101)` returns Diglett (the default pokemon)

`check_battle()` returns the result of a single Rock-Paper-Scissors encounter. Inputs are the RPS selections for the computer and player and the return value is a string that has one of these values: `"DRAW!"` (if the RPS battle is a tie), `"COMPUTER"` (if the computer RPS value defeats the player value) or `"PLAYER"` (if the player RPS value defeats the computer value)

Our auto-tester will validate these functions. Once you have completed the functions described above, finish building the rest of the game as described below:

Your program will implement a computer vs. human Pokemon tournament. You must allow the human to select which team they want to "coach" (Red or Blue). Each team will automatically select from the Pokemon list to determine which Pokemon they are playing for the current round.

Both teams use the same set of Pokemon, so it's okay if both teams field the same Pokemon for the match (e.g.: Red Jigglypuff vs. Blue Jigglypuff).

Your program should automate the selection of the Pokemon for both the human and the computer for each round. If both Pokemon selected the same RPS choice, indicate that round is a "draw" and there is no winner.

Each Pokemon is then given the opportunity to make an RPS choice. The human player specifies their RPS choice for the Pokemon they are "coaching"; your program should select a random RPS choice for the computer's Pokemon. After each round, the winner of the round is announced and your program must ask the user if they want to continue playing.

Once the tournament is finished, your program should print a message indicating who won the tournament (the human or computer) based on the number of rounds each team won.

Example play is below

```
= RESTART: /Users/keithbagley/Dropbox/NORTHEASTERN UNIVERSITY/CS5001/SPRI
/Homework/HW3/pokemon.py
What team do you want (red or blue)? blue
RED pokemon Pikachu vs. BLUE pokemon Raichu
Enter 1 for Rock, 2 for Paper, 3 for Scissors 1
Pikachu played PAPER. Raichu played ROCK
My RED team wins with Pikachu!

Play again (y/n)?y
RED pokemon Butterfree vs. BLUE pokemon Bulbasaur
Enter 1 for Rock, 2 for Paper, 3 for Scissors 2
Butterfree played PAPER. Bulbasaur played PAPER
It's a draw! No winner

Play again (y/n)?y
RED pokemon Diglett vs. BLUE pokemon Charmander
Enter 1 for Rock, 2 for Paper, 3 for Scissors 3
Diglett played ROCK. Charmander played SCISSORS
My RED team wins with Diglett!

Play again (y/n)?n
Tournament has ended!
RED team: 2      BLUE team: 0
I WIN!!!
```