

# Cognitive Modeling: Homework Assignment 1

## Technical Stack

January 17, 2025

All answers and solutions to non-programming questions should be submitted to LMS as a **legible** write-up (either fully digital or a scan). The use of LLMs (e.g., ChatGPT) is **explicitly discouraged**, unless specified otherwise. All code should be committed to and merged into the `main` branch of your team's GitHub repository, unless specified otherwise. Your LMS submissions should contain a single ZIP file named according to the pattern:

- CogModel\_Assignment[#]\_[TeamMember1Initials]\_[TeamMember2Initials]

### Problem 1: True-False Questions (4 points)

Mark all statements which are **FALSE**.

1. Stochastic models will always produce the same output  $x$  given the same input parameters  $\theta$ .
2. In psychology, replicability refers to obtaining consistent results using the same data and analysis methods, while reproducibility refers to obtaining consistent results by conducting a new study with different data under similar conditions.
3. In Python, the expression `5 + "5"` will result in a **TypeError**.
4. The `git rebase` command is used to squash commits in the history, but it cannot be used to reapply commits on top of a different base branch.
5. A detached HEAD state in Git means you are no longer on any branch and cannot commit changes until you switch back to a branch.
6. Function arguments in Python are passed by reference, meaning that modifying a mutable object within a function will also modify it outside the function scope.
7. Using the `is` operator in Python checks for value equality, similar to the `==` operator.
8. The `.gitignore` file in a Git repository is used to specify files that should not be tracked by Git and cannot be overridden by a user.

### Problem 2: Inverse vs. Forward Problems (6 points)

Provide three examples of inverse and forward problems, and discuss their relative computational difficulty.

### Problem 3: Git and GitHub (12 points)

1. Create a public GitHub repository, create and add a team logo to the `README` file, along with some basic introductory notes on why cognitive modeling is important for psychology and cognitive science. Create an `environment.yml` file and add all dependencies we have discussed so far. Then, in addition to the `main` branch, create separate branches for each of the two team members, from which you will be merging working code into the `main` branch.
2. Create a *merge conflict* (either for some of the coding exercises or a mock conflict) and resolve it.
3. Explain the differences between the following git commands

- (a) `git restore`
- (b) `git checkout`
- (c) `git reset`
- (d) `git revert`

in terms of undoing changes to a repository by providing a minimal (actual or a synthetic) example.

4. Fill in the following table:

Command	Affects Commit History?	Affects Staging Area?	Affects Working Directory?	Typical Use Case
<code>git reset</code>				
<code>git restore</code>				
<code>git rm</code>				

Table 1: Comparison of `git reset`, `git restore`, and `git rm`.

### Problem 3: Python and NumPy (6 points)

In this exercise, you will write a Python program that approximates the value of  $\pi$  using Monte Carlo approximation, which we will cover in more detail next week. Your program should generate a sequence of random points and use these points to estimate the value of  $\pi$ . The accuracy of the approximation should improve as the number of points increases. Here are some hints:

- Your program should generate random points with  $x$  and  $y$  coordinates ranging between -1 and 1. This will simulate points within a  $2 \times 2$  square that circumscribes a unit circle centered at the origin  $(0, 0)$ .
- For each generated point, determine whether it falls inside the unit circle. A point  $p = (x, y)$  is inside the circle if  $x^2 + y^2 \leq 1$ .
- Use the ratio of the number of points that fall inside the circle to the total number of generated points to approximate  $\pi$ . The formula is given by:

$$\pi \approx 4 \times (\text{number of points inside} / \text{total number of points}). \quad (1)$$

#### Problem 4: Polishing a Repository (8 points)

In this exercise, you will make some improvements to the raw mini-project provided in `seir.zip` on LMS. Note that you cannot create a Git repository inside another Git repository, so **you need to submit this separately**. In particular, you would:

1. Add an appropriate `.gitignore` file.
2. Ensure there are no hard-coded values within the functions; instead, provide function arguments with meaningful defaults.
3. Add `numpy` as a dependency and use `numpy` arrays instead of Python lists in the simulator.
4. Add a proper `environment.yml` file specifying all dependencies.
5. Add proper type hints using the `typing` module.
6. Use the `__init__.py` files to expose all functionality in the submodule Python files directly.
7. Use atomic commits and ensure your staging area is empty at the end of the exercise.