Planning a programming project

What do you want to make?

Planning a programming project

1. Decide on an idea
2. Write a description

Step 1: Write the basic idea for the project you want to make. Keep it as short as possible.

Step 2: Write the basic idea with a little more meat to it.

Step 3: Write all the basic functionality you want the project to have.

Step 4: Write the basic idea with the basic functionality together.

Step 5: Weed out the needs from the wants.

Step 6: Make a basic UML diagram of the project (If you don't know UML, skip this step)

Step 7: Begin Coding.

Step 8: Once the basic implementation is complete, go back to step 3 and substitute basic with more advanced. Repeat steps 6 and 7.

Planning a programming project

Breakout: a game where you control a paddle at the bottom of the screen, and you use it to hit a ball upwards and at angles to break bricks. The goal is to break all the bricks, and not let the ball through the ground too many times.

Planning a programming project

Genome analyser: a scientific program to perform comparative genome analysis. The goal is to construct a phylogenomic tree and a heat map scoring the presence/absence of genome encoded traits.

Write short description

10 minutes

Planning a programming project

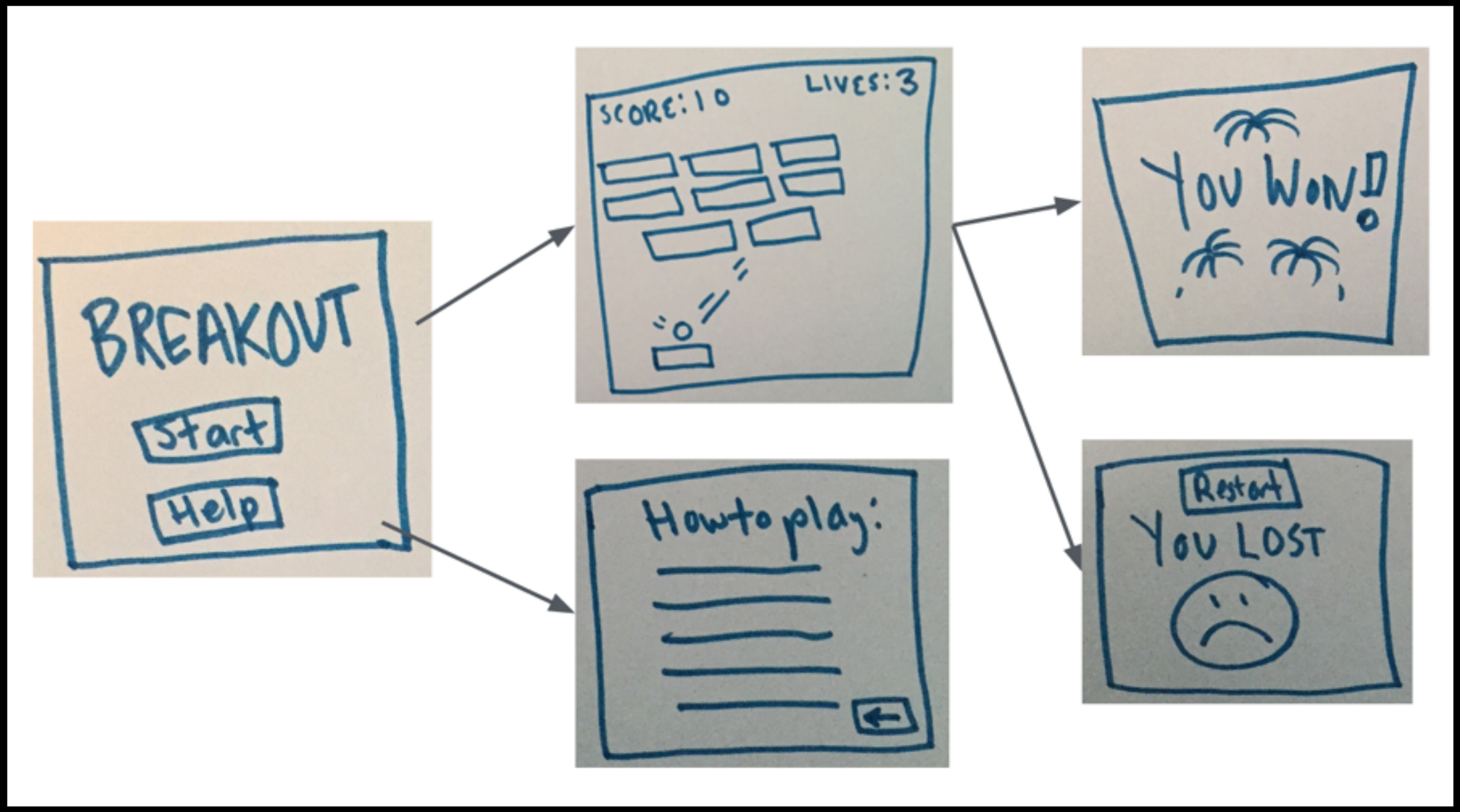
3. Decide on technology/language

Planning a programming project

3. Decide on technology/language

4. What features will the program include?

Mock ups


BREAKOUT
Start
Help
SCORE: 10
LIVES: 3
How to play:
YOU WON!!
Restart
YOU LOST

Outline of scenes

1. Game Scene
2. Main Scene
3. Help Scene
4. Win Scene
5. Lose Scene

Outline of classes

1. Genome class
2. Phlyogenomic analysis class
3. Annotation class
4. Tree class
5. Heat map class

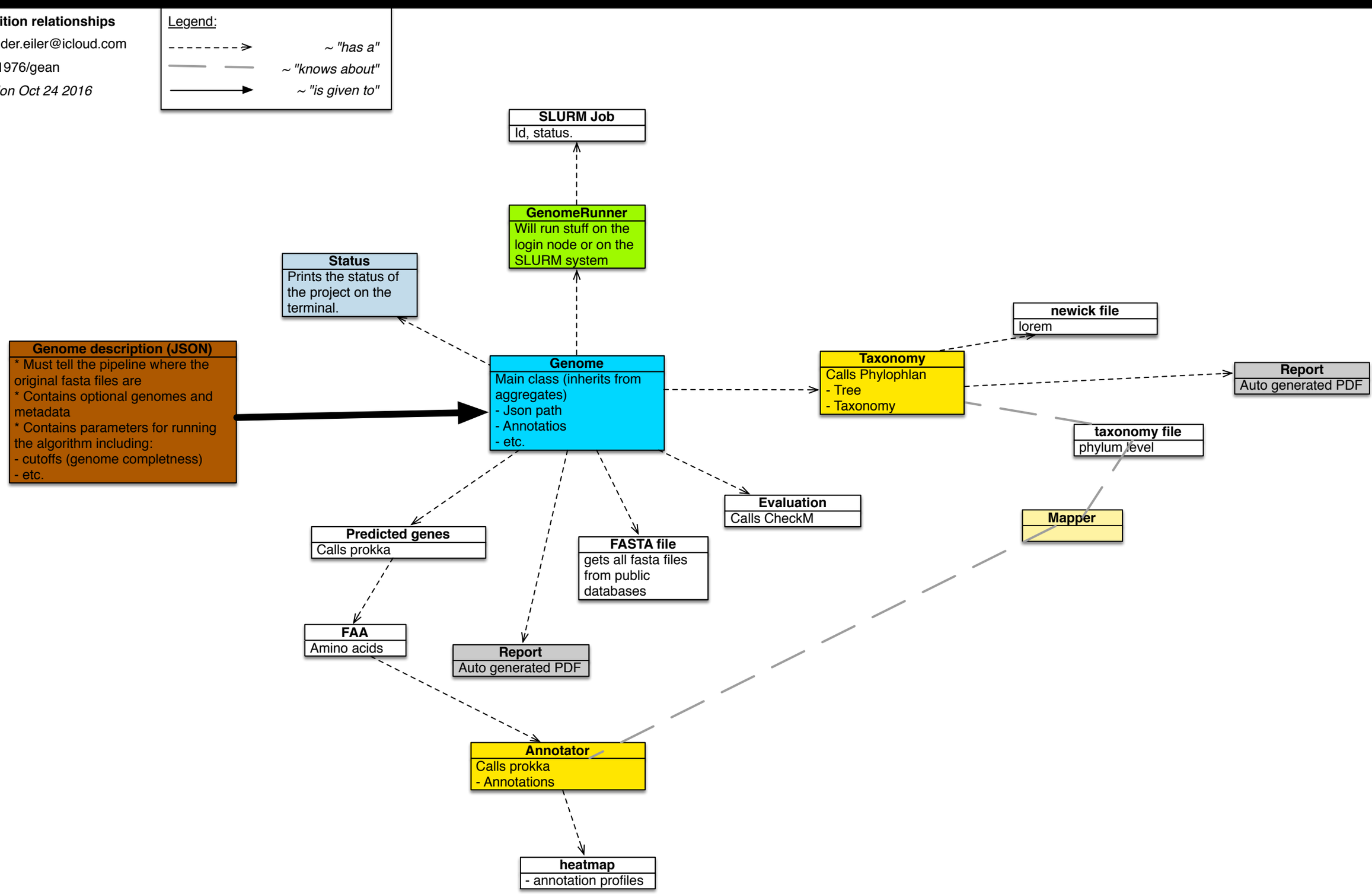
Flowchart

GEAN object composition relationships

Alexander Eielr, alexander.eiler@icloud.com

http://github.com/alper1976/gean

Last modified: *Mon Oct 24 2016*



Which features to include?

1. If I shared this with a collaborator, which features would I want to make sure were working?

2. Which features am I the most excited about?

3. Which features are the most unique to my program?

4. Which features will I learn the most from implementing?

5. Are there any features that seem too far beyond my current skill level?

How to implement it?

1. Which variables/classes should you write first?

2. Which functions?

3. Which modules are already available and can be useful?

What's the timeline?

A single week

Day 1: Design and pseudo-code
Day 2: Making individual objects and functions - testing
Day 3: Link objects and functions - testing
Day 4: Revision and polishing (write detailed documentations)

Are you ready?

Planning with pseudo code

Know what pseudocode is

Pseudocode is a step-by-step verbal outline of your code that you can gradually transcribe into programming language.

Usefulness of pseudocode

1. Describes how an algorithm should work

2. Explains a computing process/program to less technical people

3. Helps designing code in a collaborative development group.

Subjective and none-standard

Clarity is a primary goal of pseudocode, and it may help if you work within accepted programming conventions. As you develop your pseudocode into actual code, you will need to transcribe it into a programming language – so it can help to structure your outline with this in mind.

Example

A first draft of the pseudocode might look like this:

1. open the file

2. for each line in the file:
    1. look for the word
    2. remove the characters of that word
    3. insert the characters of the new word

3. then close the file.

Use iteratively

A revised draft of the pseudocode might look like this:

1. open the file
2. for each line in the file:
    1. look for the word by doing this:
        1. read character in the line
        2. if the character matches then:
            1. if all the following characters match
            2. then there is a true match
            3. remove the character of that word
            4. insert the characters of the new word
3. then close the file.

Use to add features

A revised draft of the pseudocode might look like this:

1. open the file

2. for each line in the file:
    1. look for the word by doing this:
        1. read character in the line
        2. if the character matches then:
            1. if all the following characters match
            2. then there is a true match
            3. remove the character of that word
            4. insert the characters of the new word

3. then close the file.

Some more tips

1. write only one statement per line
2. from task list to pseudocode

Tasks

```
1 read file
2 read genome by genome
3 compute number of contigs and genome size
4 write number of contigs and genome size to file
```

Pseudocode

```
1 READ infield
2 WHILE genome entry
3 COMPUTE number of contigs
4 COMPUTE genome size
5 WRITE to outfield
```

Some more tips

1. write only one statement per line
2. from task list to pseudocode
3. capitalize the initial keyword
4. write what you mean, not how to program
5. leave nothing to imagination

Some more tips

Valid pseudocode

```
If password valid then display account information.
```

```
Return leaves of tree belonging to Firmicutes.
```

Invalid pseudocode

```
let g=54/r.
```

```
do the main processing until it is done.
```

Some more tips

1. write only one statement per line
2. from task list to pseudocode
3. capitalize the initial keyword
4. write what you mean, not how to program
5. leave nothing to imagination
6. use standard programming instructions

Example

A revised draft of the pseudocode might look like this:

1. open the file

2. for each line in the file:
    1. look for the word by doing this:
        1. read character in the line
        2. if the character matches then:
            1. if all the following characters match
            2. then there is a true match
            3. remove the character of that word
            4. insert the characters of the new word

3. then close the file.

Practice

1. Would this pseudocode be understood by someone who is at least somewhat familiar with the process?
2. Is the pseudocode written in such a way that it will be easy to translated it into python?
3. Does the pseudocode describe the complete process without leaving anything out?
4. Is every object name used in the pseudocode clearly understood by the target audience?
5. Have you thought about assertions and exceptions?

Write pseudocode

10 minutes

Last

1. Read over the finished project for logic and syntax errors
2. Review the pseudocode.
3. Save your pseudocode

Translating pseudocode

1. Read once more and understand your pseudocode
2. Make sure that your actual, implemented code follows the the pseudocode.