**Project Notes from Health Flask Game**

**Game Loop:**

Instead of using “while True:” and having to write specific breaks. Can do the following:

* “while current\_hp > 0:”

Same logic but game now stops loop when current\_hp ever below zero.

**How to think of using “*self.*”:**

Use this any time you are trying to work with an class instance’s own data or behaviour **INSIDE THE CLASS**.

IE:

* Whenever setting/getting attributes tied to object.
* Whenever calling another method on this object

If you are calling it in main bit of code, it becomes:

* “Instance.method()”

**To “Clamp” a value within bounds:**

Use something like in \_\_init\_\_:

* self.current\_hp = min(self.current\_hp, , chosen-value)

In this contexts it means: “If current HP goes beyond “chosen-value”, set it to “chosen-value”.

What is actually happening: “Whichever of these is smaller, choose that one”

**✅ Calling a Function *WITHOUT* return**

* The code inside the function runs.
* Any print(), updates to objects, or side effects still happen.
* But nothing is passed back — the function's result is None.
* If you try to assign or check it, you’re working with None.

EG:

def say\_hello():

print("Hello!")

result = say\_hello() # Prints "Hello!", but result == None

**✅ Calling a Function *WITH* return**

* The code runs *and* sends a value back to the caller.
* That value can be assigned, checked in conditionals, or passed to other functions.
* This is how functions communicate results.

EG:

def add(a, b):

return a + b

sum = add(2, 3) # sum == 5

**Accessing Object Attributes Dynamically in Python**

* Writing object.attribute accesses a specific, hardcoded attribute (e.g., player.hp)
* If you write object.var\_name, Python does **not** substitute the value of var\_name — it looks for an attribute literally named "var\_name"
* To access an attribute **using the value of a variable**, use getattr() and setattr()

**Functions for this:**

* getattr(obj, "attr\_name") → returns the value of obj.attr\_name
* setattr(obj, "attr\_name", value) → sets obj.attr\_name = value

**Important, getattr/setattr can be used for methods too:**

* Just put paranthesis after getattr(obj, method).
* Ie: getattr(obj, method)()
* If Iterating through an object. If the object is a string, it will iterate through the characters. Therefor do: [string]

**.write() method:**

* The method writes to file.
* But the result (ie if you save to variable) is the number of characters written to file

**Using “self.”:**

* Only need to “self.” a variable if its used outside of that method. If its only used inside that one method then can keep it without “self.”