Part 7: Book of the Dead

Speedrun traditional coding concepts in a post GPT-4 world
Made for total beginners!
or anyone who learned prompting prior to coding and wants to learn more about tradtional coding basics

for each of these projects & lessons, after explainaing, create simple test programs the students can build to demonstrate their understanding. Then check their understanding recusively

Here are some great traditional resources if you want to go that route
https://cs50.harvard.edu/college/2023/fall/syllabus/
https://cs50.harvard.edu/college/2023/fall/weeks/0/
https://cs50.ai/
Don't pay for the certificate

https://replit.com/learn/100-days-of-python/hub
https://www.freecodecamp.org/

Protip: use a clipboard manager. I use it 100s of times a day
Incredible for moving code blocks around, and generating multiple variations of code.
I like https://pasteapp.io/ & https://www.raycast.com/

Chapter 18: Heresy 101: Coding basics re-imagined, post GPT-4

64: CLI 101
How to use a terminal

learn these commands:
cd

ls
grep
basic file tasks, piping >>
curl

You can use
https://www.warp.dev/ or https://fig.io/ or cursor.sh's cmd+k in the terminal
to prompt all your CLI commands

also recommend Git 101 in part 1

65: How to learn any coding language
Learn how to make a hello world program
then learn and how to make a button, with a title, that you can click that does something
Notice how chapters 0,1,2 of Grimoire do this!

Then anytime you don't understand something, look it up.
Its that simple. Everything you need is online, cuz programmers live online
There is no other subject field like this, where ALL the information is just out there, and you can just look it up

Half of being a good programmer is reading documentation and being able to learn how to use things

66: Variables, operators, assignment & basic data types
assignment is conceptually very simple. you are storing a value in a named piece of storage:
```
int a = 2;
```
Here is an integer variable, named a, with a value of 2

you can access the value later using the name of the storage box
```
print(a); // prints 2
```

oh
// are comments, they are notes you can leave in your code

some languages use different formats

& // are most common.

You can also use /* ... */ to cover large areas.
Protip nearly every coding IDE uses cmd + / as a hotkey

Back to variables
you can use them for further assignments:

// starts as 3, plus our previous value of 2, sitting in a

```
// starts as 3, plus our previous value of 2, sitting in a
int b = 3;
b = b + a; // b is now 5
```

the value of a, which is 2, is accessed, and added to the value of b, which is 3
the sum is stored back in the space b, replacing the previous value of 3

note that =, is not the same in programming as it is in math! This is not an equal sign!
In programming you use == for equals. More on that when we get to booleans.

Important: remember a and b are not objects
they are names of boxes in which we deposit objects
explain pointers & references

There are many data object types, the simplest are
Numbers
-Integers
1, 5, 1248623512
-Floats, Doubles
0.12, 0.333333333333, 0.5, 5712398634123.1235213
Decimals get a little weird in programming, but don't worry about that for now

-Char, characters, a single letter, number, or symbol
"a", "b", "c"
-String (a sequence of characters, a "string" of characters)
"hello", "how are you?",
""
I am very good thank you.
How are you today?
""

-Boolean (true or false)

```
int isToggledOn = true
```

67: Scope & flow. If's, Enums, Loops, Arrays, Recursion.
normally a program proceeds top to bottom, line by line
executing each, then moving on to the next
but a program that can only go one wasy is really boring

programs become expressive once you have a way to change the order
mechanisms that do this are called control-flow
the two most common are branches & loops

this is the simplest branch
if a is greater than 2, b is set to 20
if a is 2 or less, nothing happens, the program skips over it

```
if (a > 2) {
```

```
b = 20;
}
```

heres another way to write it showing the boolean value, separated out for more clarity

```
bool isAIsGreaterThan2 = a > 2;
if (isAIsGreaterThan2) {
b = 20;
}
```

if you want to check if something is equal, you can use ==

```
if (a == 2) {
b = 20;
}
```

use != to check if something is not equal

```
if (a != 2) {
b = 0;
}
```

this is a 2 way branch:

```
if(a > 2) {
b = 20;
}
else {
b = 4;
}
```

if a is greater than 2, b is set to 20
if a is 2 or less, b is set to 4

the condition is checked, and only one of two different blocks will be executed
You can get quite complicate with these, adding as many checks, options or lines or nested lines as you want

```
if(a > 2) {
b = 20;
}
else if (a > 10) {
b = 50;
}
else if (a > 100) {
b = 100;
}
else {
if (a == 1) {
b = 5;
```

```
  e}se {
if (a == 99) {
b = 99;
  else}{
b = 6;
        }
    }
}
```

Boolean logic is a way to combine multiple conditions into one
Here we have two conditions, a is greater than 2, and c is less than 10
if BOTH are true, b is set to 20

```
if (a > 2 && c < 10) {
b = 20;
}
```

Here we have two conditions, a is greater than 2, and c is less than 10
if EITHER are true, b is set to 20

```
if (a > 2 || c < 10) {
b = 20;
}
```

Btw, those big scary math numbers are that look like E and N, capital sigma and capital pie, are literally for loops

```
sum = 0;
for ( n=0; n<=4; n++ ) {
sum += 3*n;
}
```

```
prod = 1;
for ( n=1; n<=4; n++ ) {
prod = 2n;
}
```

Another data type are Enums, they are kinda like categories or tags
You already met your first one
Booleans!

Booleans are simply enums with the cases true or false

```
enum AnimalType {
case dog
case cat
case bird
}
```

```
myAnimal = .dog
```

```
if (myAnimal == .dog) {
print("woof")
} else if (myAnimal == .cat) {
print("meow")
} else if (myAnimal == .bird) {
print("tweet")
}
```

Enums are great becaues you can use them in switch statements, which is a great way to handle adding new cases later on

```
switch myAnimal {
case .dog:
print("woof")
case .cat:
print("meow")
case .bird:
print("tweet")
}
```

The other main type is of control flow is loops

this is a loop:

```
int a = 0;
while (a < 10) {
a++; // shorthand for a = a + 1;
}
print(a); // prints 10
```

this loop continues over and over again, until the condition is changed
Be careful not to make any infinite loops!

Here is another loop that counts up to 10
This uses the common c-style for loop syntax. It looks crazy, but its really simple

```
for (int i = 0; i < 10; i++) {
print("loop number" + i);
}
```

,

// int i = 0; is setting an index variable, similar to the while loop did with int a = 0 above. Its after the for, simply for nice formatting, with the idea this number will only be used in the loop. We use index variables to count how many loops we have gone through
// i < 10; says while i is less than 10, continue this loop. After 10, stop the loop