I am a grade ten student taking part in a stem based course in which we've been assigned to create a piece of code including the basic statements featured in modern day coding.

What are some ideas of 5 beginner projects I could attempt to tackle

Here are some easy coding projects perfect for complete beginners, designed to help you grasp fundamental concepts and build confidence:

1. "Hello, World!" Variations

This is the classic starting point for any programming language. It teaches you basic output and how to run a program.

2. Simple Calculator

Build a program that can perform basic arithmetic operations. This project helps you learn about input/output, variables, and conditional statements.

3. Number Guessing Game

A fun and interactive project that reinforces loops and conditional logic. The computer "thinks" of a random number, and you try to guess it.

4. To-Do List Application

A practical project that introduces data structures and basic data management. You learn how to add, remove, and display items in a list.

5. Text-Based Adventure Game

Unleash your creativity and build a simple story-driven game. This project teaches you conditional statements, loops, and string manipulation.

Could you expand on number 3 and break it down into segments and teach me in a slow manner that allows me to learn individual statements one by one.

Step 1: The Goal The goal is simple: the computer thinks of a number, and the player tries to guess it.

Step 2: The Code We'll need two things: a **secret number** and a way to get a **guess from the player**. For now, let's use a fixed number.

Python

```
None
secret_number = 7
guess = int(input("Guess a number: "))
```

Here, we've created two **variables**: `secret_number` holds the number 7, and `guess` holds whatever number the user types in. We use `int()` to make sure the user's text input is treated as a number.

Step 3: Comparing the Numbers Now we'll use an **`if` statement** to check if the guess is correct.

Python

```
None
if guess == secret_number:
    print("You win!")
```

The `==` operator checks for **equality**. If the condition is true, the indented line runs.

How do I add if and else statements to this project and do not just give me the code teach me how to implement them?

Step 1: Adding the `else` Statement The `else` statement runs its code **only if the `if` statement is false**.

Python

```
None
if guess == secret_number:
    print("You win!")
else:
    print("Sorry, that's not it.")
```

Now, the player gets feedback even if they're wrong.

Step 2: Adding a `while` Loop A `while` loop keeps running its code **as long as a condition is true**. We want the game to continue as long as the player's guess is not correct.

Python

```
None
secret_number = 7
guess = 0 # Initialize the guess

while guess != secret_number:
    guess = int(input("Guess a number: "))
    # We will add our hints here soon!
    # The loop stops when guess == secret_number

print("You win!")
```

The `!=` operator means "is not equal to". The loop will repeat until the player gets the right number.

How do I make it so that a random number is generated every time?

Step 1: Import the `random` module Python has a built-in module called `random` that contains tools for this. You need to tell your program to use it with an `import` statement at the very top.

Python

```
None
import random
```

Step 2: Use `random.randint()` The `random.randint()` function takes two numbers and returns a random whole number between them (including both numbers). We'll use it to pick our secret number.

Python

```
None
import random

secret_number = random.randint(1, 10)
```

Every time you run your program, `secret_number` will be different!

How do I make it so that it will tell you whether your guess is lower or higher than what you guesses? Don’t edit my code but break down the steps?

Here's the new core logic:

Python

```
None
import random


secret_number = random.randint(1, 10)


while True: # We'll use an infinite loop and 'break' to win
    guess = int(input("Guess a number between 1 and 10: "))


    if guess < secret_number:
        print("Too low! Try again.")
    elif guess > secret_number:
        print("Too high! Try again.")
    else:
        print("You win! The number was indeed", secret_number)
        break # This command exits the loop
```

- `while True:` creates a loop that runs forever until we use `break`.
- `elif` (short for "else if") lets us check a second condition if the first one was false.
- The `break` command is how the program knows to stop the game when the player wins.

How do I add a limit of guesses?

Step 1: Initialize a counter We'll create a variable called `guesses_used` and set it to 0 before the loop starts.

Python

```
None
guesses_used = 0
```

Step 2: Update the loop condition The loop should now continue as long as `guesses_used` is less than 3.

Python

```
None
while guesses_used < 3:
```

Step 3: Increment the counter Inside the loop, after each guess, we'll add 1 to our counter. `guesses_used += 1` is a shortcut for `guesses_used = guesses_used + 1`.

Step 4: Handle "Game Over" We'll add an `if` statement **after** the loop to check if the player lost.

Python

```
None
if guess != secret_number:

    print("Game over! You ran out of guesses.")
```

This code only runs if the loop ended because the guess limit was reached.

How do I make it show how many guesses you have left step by step

Step 1: Use `total_guesses` We'll create a variable for our limit, which makes it easier to change later.

Python

```
None
total_guesses = 3

guesses_used = 0
```

Step 2: Calculate and print remaining guesses Inside the loop, after incrementing `guesses_used`, we can calculate how many guesses are left and print it using an **f-string**.

Python

```
None
guesses_used += 1

guesses_left = total_guesses - guesses_used

print(f"You have {guesses_left} guess(es) left.")
```

The `f` before the quotes lets you easily put the value of a variable right into your text.

I want to expand my game even further and make different modes.

This is a big step! We'll use a new `if/elif/else` block at the beginning to set our game's variables.

Python

```
None
import random


print("Welcome to Will's Number Game!")

mode = input("Please select your mode: Easy or Hard? (e/h): 
").lower()
```

```
if mode == 'h':
    total_guesses = 10
    number_range = 100
    print("\nHard mode selected...")
elif mode == 'e':
    total_guesses = 3
    number_range = 10
    print("\nEasy mode selected...")
else:
    print("\nInvalid mode. Defaulting to Easy.")
    total_guesses = 3
    number_range = 10
```

This code sets the `total_guesses` and `number_range` variables before the game even starts, based on the player's choice.

How do I make another mode on top of the first 2 and give custom text for completing hard.

Here is the full, corrected code that combines all the features we've discussed. This code is robust and includes a check for beating Hard mode.

Python

```
None
import random


print("Welcome to Will's Number Game!")
```

```
mode = input("Please select your mode: Easy or Hard? (e/h): ").lower()


if mode == 'h':
    total_guesses = 5
    number_range = 100
    print("\nHard mode selected! You have 5 guesses to find a number between 1 and 100.")
elif mode == 'e':
    total_guesses = 3
    number_range = 10
    print("\nEasy mode selected! You have 3 guesses to find a number between 1 and 10.")
else:
    print("\nThat's not a mode, silly. Defaulting to Easy mode.")
    total_guesses = 3
    number_range = 10


secret_number = random.randint(1, number_range)
guesses_used = 0
guess = None # Initializes guess to prevent errors


while guesses_used < total_guesses:
    try:
```

```
        guess = int(input("Enter your guess: "))
        guesses_used += 1
        if guess == secret_number:
            break
        elif guess < secret_number:
            print("Too low!")
        else:
            print("Too high!")
        if guesses_used < total_guesses:
            guesses_left = total_guesses - guesses_used
            print(f"You have {guesses_left} guess(es) left.")
    except ValueError:
        print("That's not a valid number. Please enter a whole number.")


if guess == secret_number:
    print(f"\n🎉 You got it! The number was {secret_number} in {guesses_used} guesses.")
    if mode == 'h':
        print("\n👑 You beat hard mode!")
else:
    print("\nGame over! You ran out of guesses.")
    print(f"The correct number was {secret_number}.")
```

How do I make it so I can loop and play again without restarting the code?

The Play Again Loop

To make the game repeatable, you'll use a `while True:` statement, which creates an **infinite loop**. This loop will run forever unless you explicitly tell it to stop.

At the very end of your code, after the win/loss message is displayed, you'll add an `input()` statement to ask the player if they want to play again.

If the player's answer is not "yes," you'll use a `break` command. The `break` command immediately exits the `while True:` loop, which ends the program. If they type "yes," the loop will simply start over, and a new game will begin.

Can you expand on this?

The Play Again Loop

You can make your game repeatable by putting all of your existing code inside a **`while` loop**. This loop will run the entire game from start to finish and then ask the player if they want to go again.

1. **Wrap the Game Code:** Enclose all the main game logic in a `while True:` statement. This creates an **infinite loop** that will keep the program running until you tell it to stop.
2. **Add a `break` Condition:** At the very end of your code, after the win or lose messages are displayed, you'll ask the player if they want to play again.

Python

```
None
    # Ask the user if they want to play again
    play_again = input("\nDo you want to play again? (yes/no):
").lower()
    if play_again != 'yes':
        print("Thanks for playing! Goodbye.")
        break # This command exits the loop and ends the
program
```

If the user types anything other than "yes," the `break` command will be executed, and the program will end. Otherwise, the loop will automatically restart from the top, beginning a new game.

Thanks gemini