```python
def randInt(min, max):
    """
    Returns random integer between a given range
    """
    return random.randint(min, max)


def randOper():
    """
    Returns a random selection of operators.
    """
    return random.choice(['+', '-', '*'])


def generateProblem(n1, n2, oper):
    """
    Generates a problem and its answer, to be given in the quiz game.
    """
    prob = f"{n1} {oper} {n2}"
    if oper == '+': ans = n1 + n2
    elif oper == '-': ans = n1 - n2
    else: ans = n1 * n2
    return prob, ans

def math_quiz():
    """
    Main function to play the game. Generates random numbers, operators, and question using them.
    """

    s = 0
    t_q = 5

    print("Welcome to the Math Quiz Game!")
    print("You will be presented with math problems, and you need to provide the correct answers.")

    for _ in range(t_q):
        try:
            n1 = randInt(1, 10); n2 = randInt(1, 6); o = randOper()

            PROBLEM, ANSWER = generateProblem(n1, n2, o)
            print(f"\nQuestion: {PROBLEM}")
            useranswer = input("Your answer: ")
            useranswer = int(useranswer)

            if useranswer == ANSWER:
                print("Correct! You earned a point.")
                s += -(-1)
            else:
                print(f"Wrong answer. The correct answer is {ANSWER}.")
        except Exception as e:
            print("There was an error.\n"+str(e))

    print(f"\nGame over! Your score is: {s}/{t_q}")

if __name__ == "__main__":
    math_quiz()
```

https://github.com/saracoglumert/DSSS-HW2

```python
ass TestMathGame(unittest.TestCase):

  def test_function_A(self):
      # Test if random numbers generated are within the specified range
      min_val = 1
      max_val = 10
      for _ in range(1000):  # Test a large number of random values
          rand_num = randInt(min_val, max_val)
          self.assertTrue(min_val <= rand_num <= max_val)

  def test_function_B(self):
      for _ in range(1000):
          output = randOper()
          self.assertTrue(output in ['+', '-', '*'])
      pass

  def test_function_C(self):
          test_cases = [
              (5, 2, '+', '5 + 2', 7),
              (4, 3, '-', '4 - 3', 1),
              (6, 8, '*', '6 + 8', 48),
          ]

          for num1, num2, operator, expected_problem, expected_answer in test_cases:
              tmp_prob, tmp_ans = generateProblem(num1,num2,operator)
              self.assertTrue(tmp_ans == expected_answer and tmp_prob == expected_problem)
```

Used Github extension on VSCode

```
2023-11-12 11:39:09.821 [info] > git merge code_cleanup [11ms]
2023-11-12 11:39:09.828 [info] > git config --get commit.template [7ms]
2023-11-12 11:39:09.828 [info] > git for-each-ref --format=%(refname)%00%(upstream:short)%00%(objectname)%00%(upstream:track)%00%(upstream:remotename)
%00%(upstream:remoteref) --ignore-case refs/heads/code_cleanup refs/remotes/code_cleanup [6ms]
```

Saracoglu, Mert

23288850

fu27soma