Congratulations! You passed!
TO PASS 80% or higher

Keep Learning

GRADE
100%

Practice Quiz: While Loops

TOTAL POINTS 5

1. What are while loops in Python? **1 / 1 point**

- (•) While loops let the computer execute a set of instructions while a condition is true.
- ( ) While loops instruct the computer to execute a piece of code a set number of times.
- ( ) While loops let us branch execution on whether or not a condition is true.
- ( ) While loops are how we initialize variables in Python.

Correct
Right on! Using while loops we can keep executing the same group of instructions until the condition stops being true.

2. Fill in the blanks to make the print_prime_factors function print all the prime factors of a number. A prime factor is a number that is prime and divides another without a remainder. **1 / 1 point**

```
def print_prime_factors(number):
  # Start with two, which is the first prime
  factor = 2
  # Keep going until the factor is larger than the number
  while factor <= number:
    # Check if factor is a divisor of number
    if number % factor == 0:
      # If it is, print it and divide the original number
      print(factor)
      number = number / factor
    else:
      # If it's not, increment the factor by one
      factor += 1
  return "Done"

print_prime_factors(100)
# Should print 2,2,5,5
# DO NOT DELETE THIS COMMENT
```

Run
Reset

Correct
You nailed it! You've got the code to print all the right prime factors. Well done!

3. The following code can lead to an infinite loop. Fix the code so that it can finish successfully for all numbers. **1 / 1 point**

Note: Try running your function with the number 0 as the input, and see what you get!

```
def is_power_of_two(n):
  # Check if the number can be divided by two without a remainder
  while n != 0 and n % 2 == 0:
    n = n / 2
  # If after dividing by two the number is 1, it's a power of two
  if n == 1:
    return True
  return False


print(is_power_of_two(0)) # Should be False
print(is_power_of_two(1)) # Should be True
print(is_power_of_two(8)) # Should be True
print(is_power_of_two(9)) # Should be False
```

Run
Reset

Correct
Awesome! You fixed a tricky error that was hard to find and the function now behaves correctly.

4. Fill in the empty function so that it returns the sum of all the divisors of a number, without including it. A divisor is a number that divides into another without a remainder. **1 / 1 point**

```
def sum_divisors(n):
  sum = 0
```

```
    # Return the sum of all divisors of n, not including n
    values_to_check = round(n/2)
    while n != 0 and values_to_check != 0:
      if n % values_to_check == 0:
        sum = sum + values_to_check
      values_to_check -= 1
    return sum

print(sum_divisors(0))
# 0
print(sum_divisors(3)) # Should sum of 1
# 1
print(sum_divisors(36)) # Should sum of 1+2+3+4+6+9+12+18
# 55
print(sum_divisors(102)) # Should be sum of 2+3+6+17+34+51
# 114

```

Run

Reset

Correct

Well done, you! You've written a complex while loop and got Python to do the work for you.

5. The multiplication_table function prints the results of a number passed to it multiplied by 1 through 5. An additional requirement is that the result is not to exceed 25, which is done with the break statement. Fill in the blanks to complete the function to satisfy these conditions.

1 / 1 point

```
def multiplication_table(number):
    # Initialize the starting point of the multiplication table
    multiplier = 1
    # Only want to loop through 5
    while multiplier <= 5:
        result = number * multiplier
        # What is the additional condition to exit out of the loop?
        if result > 25 :
            break
        print(str(number) + "x" + str(multiplier) + "=" + str(result))
        # Increment the variable for the loop
        multiplier += 1

multiplication_table(3)
# Should print: 3x1=3 3x2=6 3x3=9 3x4=12 3x5=15

multiplication_table(5)
# Should print: 5x1=5 5x2=10 5x3=15 5x4=20 5x5=25

multiplication_table(8)
# Should print: 8x1=8 8x2=16 8x3=24
```

Run

Reset

Correct

Excellent! You completed the multiplication table with all of the required criteria, and it looks great!