Day 8 of Python Assignment Answers -

22. Write a python program to to check whether a number is a prime?

```python
def is_prime(num):
    """
    Check if a number is a prime number.

    Args:
        num (int): The number to be checked.

    Returns:
        bool: True if the number is prime, False otherwise.
    """
    # numbers less than or equal to 1 are not prime
    if num <= 1:
        print(num, "is not a prime number.")
    else:
        for i in range(2, int(num ** 0.5) + 1):
            # if the number is divisible by any number between 2 and sqrt(num),
            # it's not prime
            if num % i == 0:
                print(num, "is not a prime number.")
                break
        else:
            print(num, "is a prime number.")
```

Pradeepchandra Reddy S C

soopertramp07

we use print statements to output whether the number is prime or not. If the number is less than or equal to 1, we print that it is not prime.

Otherwise, we iterate through a range of numbers from 2 up to the square root of the number (inclusive), checking if the number is divisible by any of them.

If it is, we print that the number is not prime and exit the loop using the break statement. If the loop completes without finding a divisor, we print that the number is prime using the else clause that is executed after the loop completes successfully.

Pradeepchandra Reddy S C

soopertramp07

The reason we use the square root of num in the range of the loop is to optimize the algorithm and reduce the number of computations needed to check if a number is prime.

Suppose we have a number num that is not prime. Then there must be two factors of num, say a and b, such that num = a * b. One of a and b must be less than or equal to the square root of num, and the other must be greater than or equal to the square root of num. Otherwise, their product would be greater than num.

For example, suppose num = 24. The factors of num are 1, 2, 3, 4, 6, 8, 12, 24. Notice that the largest factor less than or equal to the square root of num (which is 4.89897948557...) is 4, and the smallest factor greater than or equal to the square root of num is 6. So if we check all the factors of num up to 4, we can be sure that if none of them divide num evenly, then num is not prime.

Therefore, we only need to check for factors of num up to the square root of num. This reduces the number of computations required and makes the algorithm more efficient.

Pradeepchandra Reddy S C

soopertramp07

23. Write a Python program to print all prime numbers in the interval Of 1 - 10000?

```python
def print_primes(start=1, end=10000):
    """
    Prints all prime numbers within the given interval.

    Args:
        start (int): The starting number of the interval (default is 1).
        end (int): The ending number of the interval (default is 10000).

    Returns:
        None.
    """
    for num in range(start, end + 1):
        # check if the number is greater than 1
        if num > 1:
            # check if the number is prime
            for i in range(2, int(num ** 0.5) + 1):
                if num % i == 0:
                    break
            else:
                print(num, end=' ')
```

Pradeepchandra Reddy S C

soopertramp07

The print_primes function takes two optional arguments, start and end, which specify the range of numbers to check for primes.

The function then iterates through each number in that range, checking whether it is prime or not.

If the number is greater than 1, it checks if it is prime by dividing it by all numbers from 2 to the square root of the number.

If the number is divisible by any of those numbers, it is not prime, and the loop exits using the break statement.

If the loop completes without finding a divisor, the number is prime, and it is printed to the console using the print statement.

Pradeepchandra Reddy S C

soopertramp07

24. Write a Python program to find the factorial of a number

```python
def factorial(n):
    """
    Compute the factorial of a given integer.

    Args:
        n (int): The number whose factorial to compute.

    Returns:
        int: The factorial of n.

    Raises:
        ValueError: If n is negative.
    """
    # Check that n is non-negative
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")

    # Base case: factorial of 0 is 1
    if n == 0:
        return 1

    # Recursive case: compute factorial of n-1 and multiply by n
    else:
        return n * factorial(n-1)
```

Pradeepchandra Reddy S C

soopertramp07

The factorial function takes an integer n as input and returns its factorial as an integer.

The function first checks that the input is non-negative, raising a ValueError if it is not.

The base case is that the factorial of 0 is 1, which is returned immediately.

For all other values of n, the function computes the factorial of n-1 recursively and multiplies the result by n. This continues until the base case is reached.

Pradeepchandra Reddy S C

soopertramp07