1. Write a Python program to check if the given number is a Disarium Number?

ANS: A Disarium Number is a number that is equal to the sum of its digits, each raised to the power of its respective position (1-indexed). For example, 89 is a Disarium Number because 8^1 + 9^2 = 8 + 81 = 89.

To check if a given number is a Disarium Number, we need to calculate the sum of its digits, each raised to the power of its respective position, and then compare it with the original number. If they are equal, the number is a Disarium Number. Here's a Python program to do that:

def is\_disarium\_number(number):

num\_str = str(number)

length = len(num\_str)

total = sum(int(digit) \*\* (index + 1) for index, digit in enumerate(num\_str))

return number == total

def main():

number = int(input("Enter a number to check if it's a Disarium Number: "))

if is\_disarium\_number(number):

print(f"{number} is a Disarium Number.")

else:

print(f"{number} is not a Disarium Number.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

In this program, the `is\_disarium\_number()` function takes a number as input and converts it to a string to iterate over its digits. It then calculates the sum of each digit raised to the power of its respective position using a list comprehension and the `enumerate()` function. Finally, it checks if the sum is equal to the original number and returns `True` or `False` accordingly.When we run the program, it will prompt you to enter a number, and it will then tell you if the entered number is a Disarium Number or not.

1. Write a Python program to print all disarium numbers between 1 to 100?

ANS: def is\_disarium\_number(number):

num\_str = str(number)

length = len(num\_str)

total = sum(int(digit) \*\* (index + 1) for index, digit in enumerate(num\_str))

return number == total

def print\_disarium\_numbers(start, end):

disarium\_numbers = [num for num in range(start, end + 1) if is\_disarium\_number(num)]

return disarium\_numbers

def main():

start = 1

end = 100

disarium\_numbers = print\_disarium\_numbers(start, end)

if disarium\_numbers:

print("Disarium numbers between 1 and 100:")

print(disarium\_numbers)

else:

print("No Disarium numbers found between 1 and 100.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. Write a Python program to check if the given number is Happy Number?

ANS: A Happy Number is a number that, when repeatedly replaced by the sum of the squares of its digits, eventually reaches 1. If a number ends up in an endless cycle without reaching 1, it is not a Happy Number. For example, 19 is a Happy Number because:

1^2 + 9^2 = 1 + 81 = 82

8^2 + 2^2 = 64 + 4 = 68

6^2 + 8^2 = 36 + 64 = 100

1^2 + 0^2 + 0^2 = 1 + 0 + 0 = 1

To check if a given number is a Happy Number, we can use the Floyd's cycle detection algorithm (also known as the "tortoise and hare" algorithm) to detect cycles in the sequence of numbers. If the algorithm reaches 1, the number is a Happy Number; otherwise, it's not. Here's a Python program to determine if a given number is a Happy Number:

def square\_sum\_of\_digits(number):

return sum(int(digit) \*\* 2 for digit in str(number))

def is\_happy\_number(number):

slow = number

fast = square\_sum\_of\_digits(number)

while slow != fast:

slow = square\_sum\_of\_digits(slow)

fast = square\_sum\_of\_digits(square\_sum\_of\_digits(fast))

return slow == 1

def main():

number = int(input("Enter a number to check if it's a Happy Number: "))

if is\_happy\_number(number):

print(f"{number} is a Happy Number.")

else:

print(f"{number} is not a Happy Number.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

In this program, the `square\_sum\_of\_digits()` function calculates the sum of the squares of the digits of a number. The `is\_happy\_number()` function uses the Floyd's cycle detection algorithm to check if a number is a Happy Number.

1. Write a Python program to print all happy numbers between 1 and 100?

ANS: def square\_sum\_of\_digits(number):

return sum(int(digit) \*\* 2 for digit in str(number))

def is\_happy\_number(number):

slow = number

fast = square\_sum\_of\_digits(number)

while slow != fast:

slow = square\_sum\_of\_digits(slow)

fast = square\_sum\_of\_digits(square\_sum\_of\_digits(fast))

return slow == 1

def print\_happy\_numbers(start, end):

happy\_numbers = [num for num in range(start, end + 1) if is\_happy\_number(num)]

return happy\_numbers

def main():

start = 1

end = 100

happy\_numbers = print\_happy\_numbers(start, end)

if happy\_numbers:

print("Happy numbers between 1 and 100:")

print(happy\_numbers)

else:

print("No Happy numbers found between 1 and 100.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. Write a Python program to determine whether the given number is a Harshad Number?

ANS: A Harshad Number (also called a Niven Number) is a number that is divisible by the sum of its digits. In other words, the number is evenly divisible by the sum of its individual digits. For example, 18 is a Harshad Number because 1 + 8 = 9, and 18 is divisible by 9.

To determine if a given number is a Harshad Number, we need to calculate the sum of its digits and check if the number is divisible by this sum. If it is, then the number is a Harshad Number. Here's a Python program to do that:

def sum\_of\_digits(number):

return sum(int(digit) for digit in str(number))

def is\_harshad\_number(number):

return number % sum\_of\_digits(number) == 0

def main():

number = int(input("Enter a number to check if it's a Harshad Number: "))

if is\_harshad\_number(number):

print(f"{number} is a Harshad Number.")

else:

print(f"{number} is not a Harshad Number.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

```

In this program, the `sum\_of\_digits()` function calculates the sum of the digits of a number by converting it to a string and iterating over its individual characters. The `is\_harshad\_number()` function uses the `sum\_of\_digits()` function to calculate the sum of the digits and checks if the given number is divisible by this sum.

1. Write a Python program to print all pronic numbers between 1 and 100?

ANS: A pronic number (also known as oblong number or rectangular number) is a number that is the product of two consecutive integers. In other words, a number "n" is pronic if it can be expressed as n = m \* (m + 1) for some integer "m". For example, 2, 6, 12, 20, etc., are pronic numbers because they can be expressed as the product of two consecutive integers (e.g., 2 = 1 \* (1 + 1), 6 = 2 \* (2 + 1), 12 = 3 \* (3 + 1), etc.).

To find all pronic numbers between 1 and 100, we can iterate through the range and check if each number can be expressed as the product of two consecutive integers. Here's a Python program to do that:

def is\_pronic\_number(number):

for m in range(1, number):

if m \* (m + 1) == number:

return True

return False

def print\_pronic\_numbers(start, end):

pronic\_numbers = [num for num in range(start, end + 1) if is\_pronic\_number(num)]

return pronic\_numbers

def main():

start = 1

end = 100

pronic\_numbers = print\_pronic\_numbers(start, end)

if pronic\_numbers:

print("Pronic numbers between 1 and 100:")

print(pronic\_numbers)

else:

print("No Pronic numbers found between 1 and 100.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

In this program, the `is\_pronic\_number()` function checks if a given number is a pronic number by iterating through the range from 1 to the number and checking if any "m" and "m + 1" can produce the given number.The `print\_pronic\_numbers()` function generates a list of pronic numbers in the range between `start` and `end` using a list comprehension.

The `main()` function sets the range from 1 to 100, calls `print\_pronic\_numbers()` to find the pronic numbers within that range, and then prints the list of pronic numbers if any are found.When you run the program, it will print all the pronic numbers between 1 and 100, if there are any.