This notebook contains Python programs (41–50) focused on number theory and basic string/list operations.

- It includes checks for special numbers (like Disarium, Happy, Harshad, Pronic) and basic list manipulations.
- this notebook can be accessed using (https://github.com/amitvsuryavanshi04/SIC_programming_and_coding)

Program 41 to remove punctuation from a string.

```
# define punctuation
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
# To take input from the user
my_str = input("Enter a string: ")
# remove punctuation from the string
no_punct = ""
for char in my_str:
  if char not in punctuations:
    no_punct = no_punct + char
# display the unpunctuated string
print(no_punct)
```

```
Enter a string: afniaiuagakb %&^%%*(( amit *&(
afniaiuagakb  amit
```

Program 42 to check if given number is Disarium number A Disarium number is a number that is equal to the sum of its digits each raised to the power of its respective position. For example, 89 is a Disarium number because 81 + 92 = 8 + 81 = 89.

```
def is_disarium(number):
    # Convert the number to a string to iterate over its digits
    num_str = str(number)
    # Calculate the sum of digits raised to their respective positions
    digit_sum = sum(int(i) ** (index + 1) for index, i in enumerate(num_str))
    # Check if the sum is equal to the original number
    return digit_sum == number

# Input a number from the user
try:
    num = int(input("Enter a number: "))
    # Check if it's a Disarium number
```

```
    if is_disarium(num):
        print(f"{num} is a Disarium number.")
    else:
        print(f"{num} is not a Disarium number.")
except ValueError:
    print("Invalid input. Please enter a valid number.")
```

```
Enter a number: 445
445 is not a Disarium number.
```

Program 43 to print all disarium numbers between 1 to 100

```
def is_disarium(num):
    num_str = str(num)
    digit_sum = sum(int(i) ** (index + 1) for index, i in enumerate(num_str))
    return num == digit_sum

# Generate Disarium numbers between 1 and 100
disarium_numbers = [num for num in range(1, 101) if is_disarium(num)]

print("Disarium numbers between 1 and 100:")
for num in disarium_numbers:
    print(num, end=" | ")
```

```
Disarium numbers between 1 and 100:
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 89 |
```

Program 44 to check given number is happy number

Happy Number: A Happy Number is a positive integer that, when you repeatedly replace

the number by the sum of the squares of its digits and continue the process, eventually reaches 1. If the process never reaches 1 but instead loops endlessly in a cycle, the number is not a Happy Number.

For example:

19 is a Happy Number because:

The process reaches 1, so 19 is a Happy Number

```
def is_happy_number(num):
  seen = set() # To store previously seen numbers
  while num != 1 and num not in seen:
    seen.add(num)
    num = sum(int(i) ** 2 for i in str(num))
  return num == 1
# Test the function with a number
num = int(input("Enter a number: "))
if is_happy_number(num):
  print(f"{num} is a Happy Number")
else:
  print(f"{num} is not a Happy Number")
```

```
Enter a number: 19
19 is a Happy Number
```

Program 45 Print all happy numbers between 1 and 100.

```
def is_happy_number(num):
  seen = set()
  while num != 1 and num not in seen:
    seen.add(num)
    num = sum(int(i) ** 2 for i in str(num))
  return num == 1
happy_numbers = []
for num in range(1, 101):
  if is_happy_number(num):
    happy_numbers.append(num)
print("Happy Numbers between 1 and 100:")
print(happy_numbers)
```

```
Happy Numbers between 1 and 100:
[1, 7, 10, 13, 19, 23, 28, 31, 32, 44, 49, 68, 70, 79, 82, 86, 91, 94, 97, 100]
```

Program 46 to determine whether the given number is a Harshad Number. A Harshad number (or Niven number) is an integer that is divisible by the sum of its digits. In other words, a number is considered a Harshad number if it can be evenly divided by the sum of its own digits. For example: 18 is a Harshad number because , and 18 is divisible by 9 42 is not a Harshad number because , and 42 is not divisible by 6.

```
def is_harshad_number(num):
# Calculate the sum of the digits of the number
  digit_sum = sum(int(i) for i in str(num))
# Check if the number is divisible by the sum of its digits
  return num % digit_sum == 0
# Input a number
num = int(input("Enter a number: "))
# Check if it's a Harshad Number
if is_harshad_number(num):
  print(f"{num} is a Harshad Number.")
else:
  print(f"{num} is not a Harshad Number.")
```

```
Enter a number: 20
20 is a Harshad Number.
```

Program 47

- to print all pronic numbers between 1 and 100.
- A pronic number, also known as an oblong number or rectangular number, is a type of
- figurate number that represents a rectangle. It is the product of two consecutive integers, n and (n + 1).
- Mathematically, a pronic number can be expressed as: For example, the first few pronic numbers are:

1. $Pn = n * (n + 1)$
2. $P1 = 1 * (1 + 1) = 2$
3. $P2 = 2 * (2 + 1) = 6$
4. $P3 = 3 * (3 + 1) = 12$
5. $P4 = 4 * (4 + 1) = 20$

```
def is_pronic_number(num):
  for n in range(1, int(num**0.5) + 1):
    if n * (n + 1) == num:
```

```
      return True
  return False
print("Pronic numbers between 1 and 100 are:")
for i in range(1, 101):
  if is_pronic_number(i):
    print(i, end=" | ")
```

```
Pronic numbers between 1 and 100 are:
2 | 6 | 12 | 20 | 30 | 42 | 56 | 72 | 90 |
```

Program 48 sum of elements in an array.

```
# Sample list of numbers
numbers = [10, 20, 30, 40, 50]
# Initialize a variable to store the sum
sum_of_numbers = 0
# Iterate through the list and accumulate the sum
for i in numbers:
  sum_of_numbers += i
# Print the sum
print("Sum of elements in the list:", sum_of_numbers)
```

```
Sum of elements in the list: 150
```

Program 49 multiply all numbers in the list

```
# Sample list of numbers
numbers = [10, 20, 30, 40, 50]
# Initialize a variable to store the product
product_of_numbers = 1
# Iterate through the list and accumulate the product
for i in numbers:
  product_of_numbers *= i
# Print the product
print("Product of elements in the list:", product_of_numbers)
```

```
Product of elements in the list: 12000000
```

Program 50 to find the smallest number in the list

```
# Sample list of numbers
numbers = [30, 10, -45, 5, 20]
# Initialize a variable to store the minimum value, initially set to th
minimum = numbers[0]
# Iterate through the list and update the minimum value if a smaller nu
for i in numbers:
  if i < minimum:
    minimum = i
# Print the minimum value
print("The smallest number in the list is:", minimum)
```

```
The smallest number in the list is: -45
```