

## Answer Sheet - Python

Name: Isha Garg

Phone No.: 7053017594

Email: ishagarg989@gmail.com

### Major Question 1

15 Marks

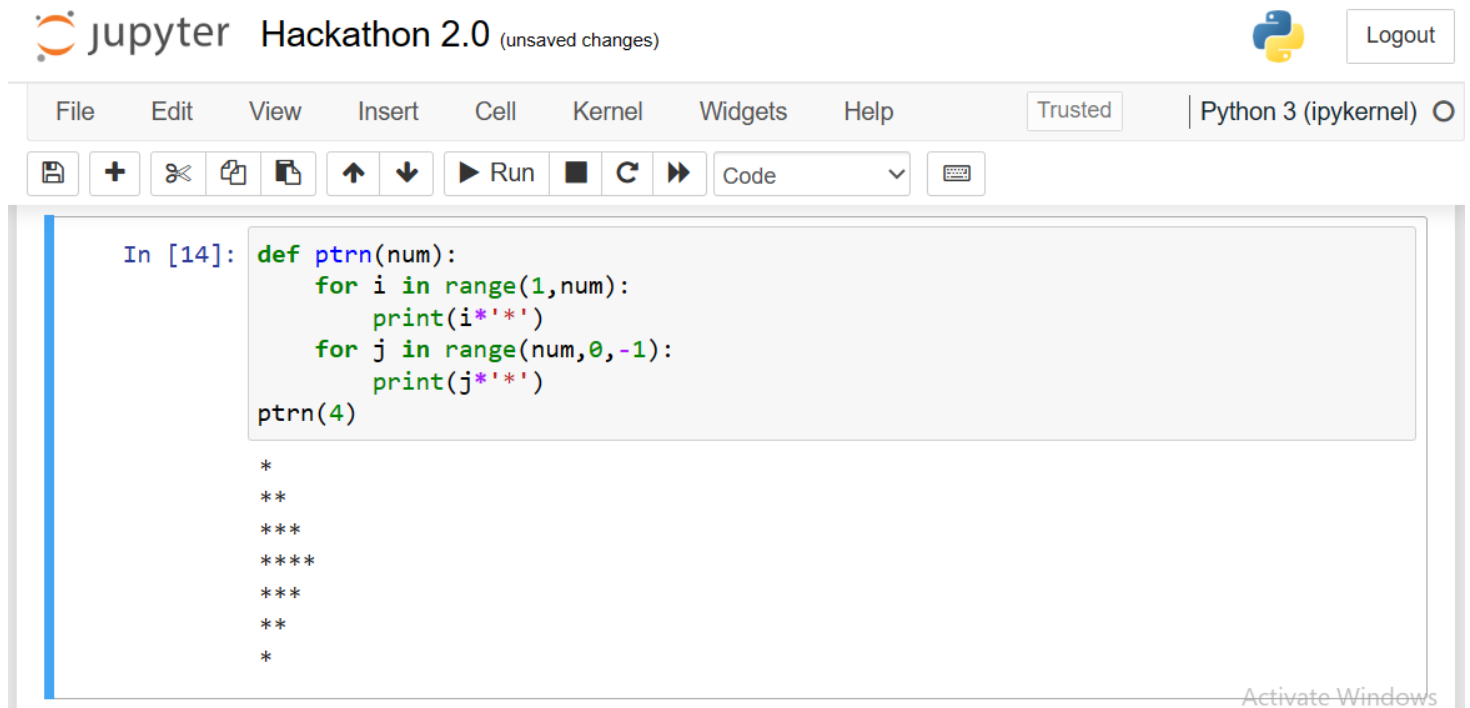
A) Write a program to print the following pattern:

```
*  
  
* *  
  
* * *  
  
* * * *  
  
* * *  
  
* *  
  
*
```

**Code:**

```
def ptrn(num):  
    for i in range(1,num):  
        print(i***)  
    for j in range(num,0,-1):  
        print(j***)  
ptrn(4)
```

## Output:



The image shows a Jupyter Hackathon 2.0 interface. At the top, there's a header with the Jupyter logo, the text "Hackathon 2.0 (unsaved changes)", a Python logo, and a "Logout" button. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3 (ipykernel)" buttons. Below the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and running code. The main area contains a code cell with the following Python code:

```
In [14]: def ptrn(num):  
          for i in range(1,num):  
              print(i*' '*')  
          for j in range(num,0,-1):  
              print(j*' '*')  
          ptrn(4)
```

The output of the code is a pattern of asterisks:

```
*  
**  
***  
****  
***  
**  
*
```

At the bottom right of the interface, there is a watermark that says "Activate Windows".

B) Write a program to accept 5 even and 5 odd numbers from the user and display

- *sum of even numbers,*
- *product of odd numbers*
- *absolute difference of the sum and product.*

*Check if the final result is a prime number or not.*

## Code:

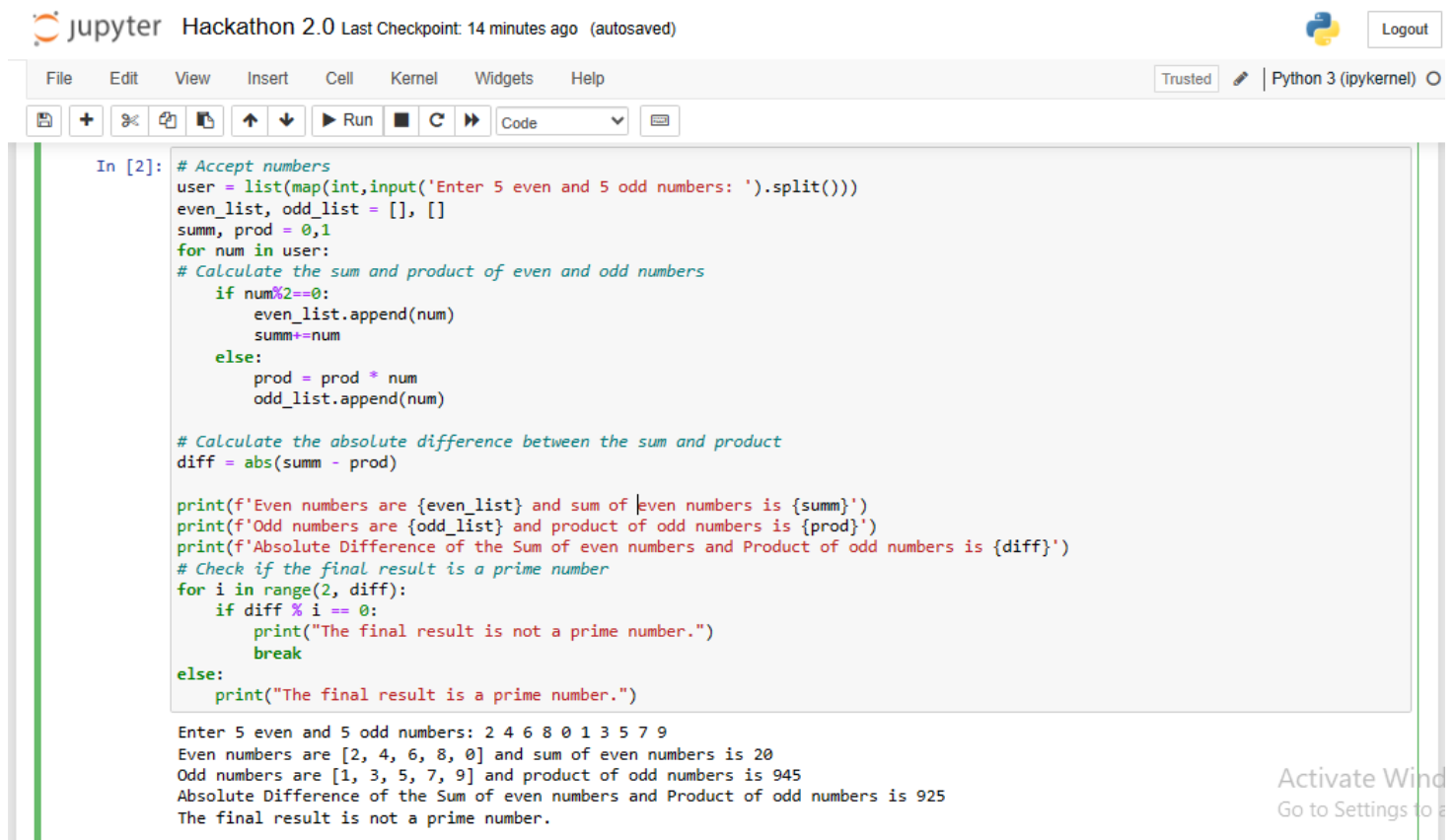
```
# Accept numbers
user = list(map(int,input('Enter 5 even and 5 odd numbers: ').split()))
even_list, odd_list = [], []
summ, prod = 0,1
for num in user:
# Calculate the sum and product of even and odd numbers
    if num%2==0:
        even_list.append(num)
        summ+=num
    else:
        prod = prod * num
        odd_list.append(num)

# Calculate the absolute difference between the sum and product
diff = abs(summ - prod)

print(f'Even numbers are {even_list} and sum of even numbers is {summ}')
print(f'Odd numbers are {odd_list} and product of odd numbers is {prod}')
print(f'Absolute Difference of the Sum of even numbers and Product of odd numbers is {diff}')

# Check if the final result is a prime number
for i in range(2, diff):
    if diff % i == 0:
        print("The final result is not a prime number.")
        break
else:
    print("The final result is a prime number.")
```

## Output:



The image shows a Jupyter Notebook interface. At the top, it says "jupyter Hackathon 2.0 Last Checkpoint: 14 minutes ago (autosaved)". The interface includes a menu bar with File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu bar is a toolbar with icons for saving, running, and other actions. The main area contains a code cell with the following Python code:

```
In [2]: # Accept numbers
user = list(map(int, input('Enter 5 even and 5 odd numbers: ').split()))
even_list, odd_list = [], []
summ, prod = 0, 1
for num in user:
    # Calculate the sum and product of even and odd numbers
    if num % 2 == 0:
        even_list.append(num)
        summ += num
    else:
        prod = prod * num
        odd_list.append(num)

# Calculate the absolute difference between the sum and product
diff = abs(summ - prod)

print(f'Even numbers are {even_list} and sum of even numbers is {summ}')
print(f'Odd numbers are {odd_list} and product of odd numbers is {prod}')
print(f'Absolute Difference of the Sum of even numbers and Product of odd numbers is {diff}')
# Check if the final result is a prime number
for i in range(2, diff):
    if diff % i == 0:
        print("The final result is not a prime number.")
        break
else:
    print("The final result is a prime number.")
```

The output of the code is as follows:

```
Enter 5 even and 5 odd numbers: 2 4 6 8 0 1 3 5 7 9
Even numbers are [2, 4, 6, 8, 0] and sum of even numbers is 20
Odd numbers are [1, 3, 5, 7, 9] and product of odd numbers is 945
Absolute Difference of the Sum of even numbers and Product of odd numbers is 925
The final result is not a prime number.
```

C) Create a class named Item that holds data about an item in a retail store. The class should have the following three properties:

- *name*: the name property is a String object that holds the name of the item.
- *price*: the price property is a double variable that holds the item's retail price
- *quantity*: the quantity property is an int variable that holds the number of units currently in inventory

Write four methods to retrieve the values from the three fields and their current inventory value

- *getName()* returns the item name String
- *getPrice()* returns the price of the item double
- *getQuantity()* returns the number of quantities int
- *getValue()* that returns the current inventory value (quantity \* price) double

## Code:

```
class Item:

    def __init__(self, name, price, quantity):

        self.name = name

        self.price = price

        self.quantity = quantity


    def getName(self):

        return self.name


    def getPrice(self):

        return self.price




    def getQuantity(self):


        return self.quantity















    def getValue(self):

        return self.quantity * self.price
```

## Output:

 **jupyter** Hackathon 2.0 Last Checkpoint: 25 minutes ago (unsaved changes)  [Logout](#)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) 

        Run    Code  

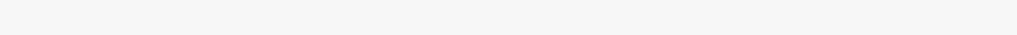
```
In [15]: class Item:
          def __init__(self, name, price, quantity):
              self.name = name
              self.price = price
              self.quantity = quantity

          def getName(self):
              return self.name

          def getPrice(self):
              return self.price

          def getQuantity(self):
              return self.quantity

          def getValue(self):
              return self.quantity * self.price
```

In [ ]: 

Activate Windows  
Go to Settings to activate Windows

A) Ask the user number of rows to be generated of a series. Suppose user enters no. of rows = 5 then the series shall be:

9

99

999

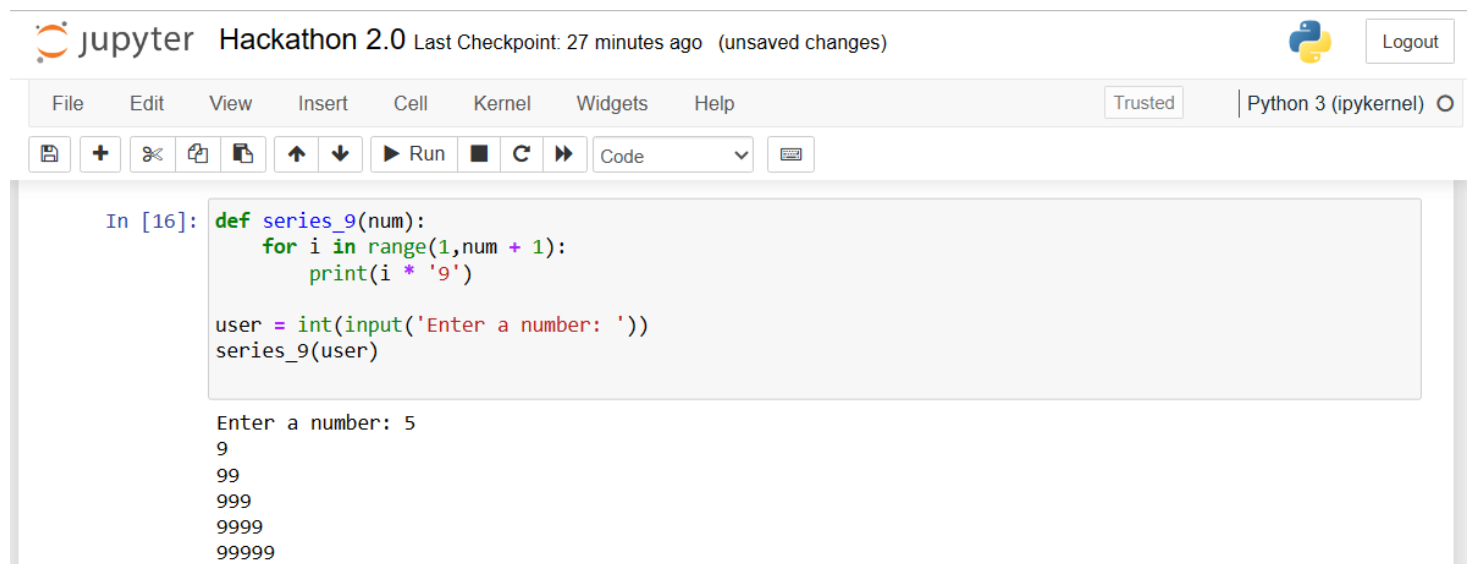
9999

99999

### Code:

```
def series_9(num):  
    for i in range(1,num + 1):  
        print(i * '9')  
  
user = int(input('Enter a number: '))  
series_9(user)
```

### Output:



The image shows a Jupyter Notebook interface. At the top, it says "jupyter Hackathon 2.0" and "Last Checkpoint: 27 minutes ago (unsaved changes)". There is a "Logout" button. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3 (ipykernel)". Below the menu bar is a toolbar with icons for saving, adding cells, undo, redo, running, and other actions. The main area shows a code cell with the following code:

```
In [16]: def series_9(num):  
          for i in range(1,num + 1):  
              print(i * '9')  
  
          user = int(input('Enter a number: '))  
          series_9(user)
```

The output of the code is:

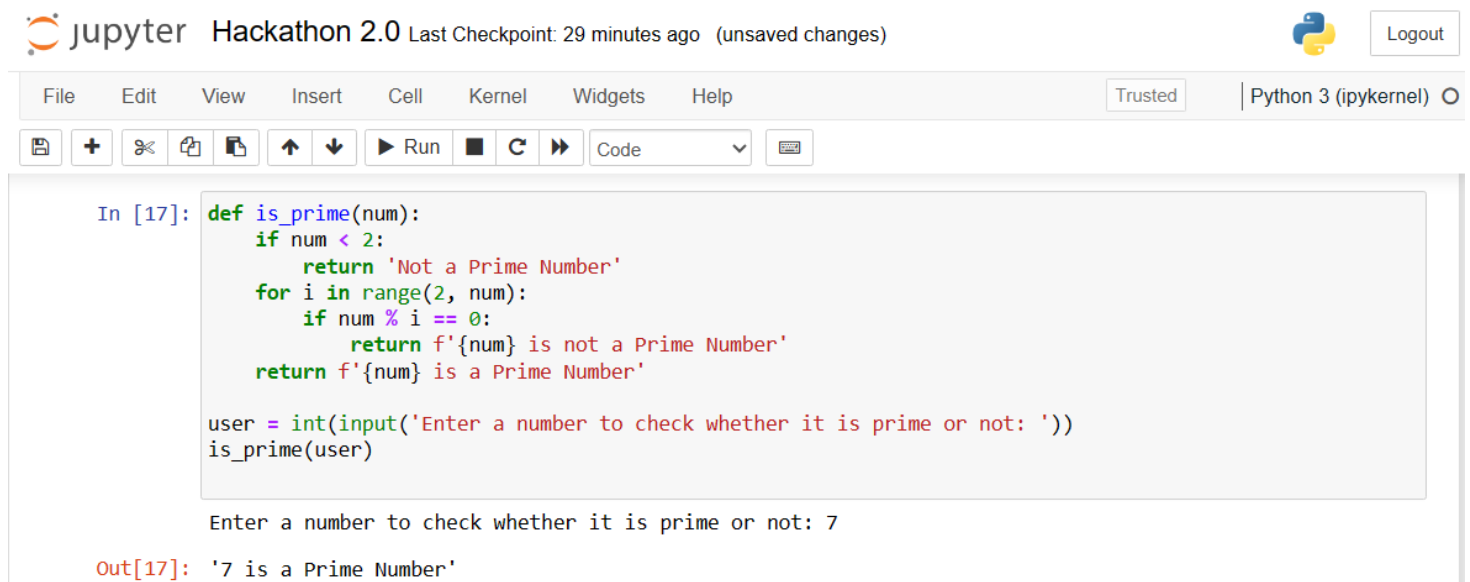
```
Enter a number: 5  
9  
99  
999  
9999  
99999
```

B) Write a program to accept a number from the user and check whether the number entered is prime or not.

### Code:

```
def is_prime(num):  
    if num < 2:  
        return 'Not a Prime Number'  
    for i in range(2, num):  
        if num % i == 0:  
            return f'{num} is not a Prime Number'  
    return f'{num} is a Prime Number'  
  
user = int(input('Enter a number to check whether it is prime or not: '))  
is_prime(user)
```

### Output:



The image shows a Jupyter Notebook interface. At the top, it says "jupyter Hackathon 2.0" and "Last Checkpoint: 29 minutes ago (unsaved changes)". There is a "Logout" button. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3 (ipykernel)". Below the menu bar is a toolbar with icons for saving, adding cells, undo, redo, and running code. The main area shows a code cell with the following code:

```
In [17]: def is_prime(num):  
         if num < 2:  
             return 'Not a Prime Number'  
         for i in range(2, num):  
             if num % i == 0:  
                 return f'{num} is not a Prime Number'  
         return f'{num} is a Prime Number'  
  
         user = int(input('Enter a number to check whether it is prime or not: '))  
         is_prime(user)
```

Below the code cell, the output is displayed:

```
Enter a number to check whether it is prime or not: 7  
Out[17]: '7 is a Prime Number'
```

C) Continued from Major Question 1. Write a separate class called Inventory with methods

- *generate()* - creates three Item objects
- *getDetails()* - produces a neatly formatted table of the store's inventory displaying the three items, their current inventory value, and the total inventory value for the store.

Name	Price	Quantity	Value
Stapler	2.25	15	33.75
Paper	32.99	255	8412.45
Binder	4.75	9	42.75

Total inventory is 8488.95

### Code:

```
class Inventory:
    def __init__(self):
        self.items = []

    def generate(self):
        # Create three Item objects
        item1 = Item("Stapler", 2.25, 15)
        item2 = Item("Paper", 32.99, 255)
        item3 = Item("Binder", 4.75, 9)

        self.items = [item1, item2, item3]
```



```
def getDetails(self):
    total_value = 0

    # Print the table header
    print("Name\tPrice\tQuantity\tValue")
    print("=====")

    for item in self.items:
        name = item.getName()
        price = item.getPrice()
        quantity = item.getQuantity()
        value = item.getValue()

        # Print item details
        print(f"{name}\t{price}\t{quantity}\t{value}")

        total_value += value

    # Print total inventory value
    print(f"Total inventory is {total_value}")
```

```
# Test the classes
inventory = Inventory()
inventory.generate()
inventory.getDetails()
```

## Output:

```
In [18]: class Inventory:
def __init__(self):
    self.items = []

def generate(self):
    # Create three Item objects
    item1 = Item("Stapler", 2.25, 15)
    item2 = Item("Paper", 32.99, 255)
    item3 = Item("Binder", 4.75, 9)

    self.items = [item1, item2, item3]

def getDetails(self):
    total_value = 0

    # Print the table header
    print("Name\t\tPrice\t\tQuantity\t\tValue")
    print("=====")

    for item in self.items:
        name = item.getName()
        price = item.getPrice()
        quantity = item.getQuantity()
        value = item.getValue()

        # Print item details
        print(f"{name}\t\t{price}\t\t{quantity}\t\t{value}")

        total_value += value

    # Print total inventory value
    print(f"Total inventory is {total_value}")

# Test the classes
inventory = Inventory()
inventory.generate()
inventory.getDetails()
```

Name	Price	Quantity	Value
Stapler	2.25	15	33.75
Paper	32.99	255	8412.45
Binder	4.75	9	42.75
Total inventory is 8488.95			