NAAN MUDHALVAN :IBM
PHASE-III
TECHNOLOGY:DATA SCIENCE
PROJECT:COVID 19 VACCINE ANALYSIS
DEVELOPMENT PART 1

INTRODUCTION

In early 2020, the SARS-CoV-2 (COVID-19) pandemic unmasked the many flaws that healthcare systems faced worldwide. While some of these issues were difficult to predict, such as the feasibility of pandemic response protocols or federal government regulations to be activated [1], other healthcare issues were to be expected, especially in the United States.

VACCINE PLANNING

Creating a comprehensive Python code for vaccine planning involves various components such as data processing, prioritization, scheduling, and tracking. Below is a simplified example demonstrating how you might structure a Python code for vaccine planning:

PROGRAM

```
import heapq


# Sample data (age and risk factor)
population_data = {
    'Alice': {'age': 30, 'risk_factor': 'high'},
    'Bob': {'age': 65, 'risk_factor': 'medium'},
    'Charlie': {'age': 40, 'risk_factor': 'low'},
    # ... Add more data
}


# Define a priority function for the heapq
def priority(person):
    age = population_data[person]['age']
    risk_factor = population_data[person]['risk_factor']
    priority_map = {'low': 1, 'medium': 2, 'high': 3}
    return (priority_map[risk_factor], -age)  # Higher risk and lower age should
be prioritized


# Prioritize individuals based on age and risk factor
priority_queue = []
for person in population_data:
    heapq.heappush(priority_queue, (priority(person), person))


# Allocate vaccine doses based on priority
```

```
vaccine_doses = 1000

allocation = {}

while vaccine_doses > 0 and priority_queue:

    _, person = heapq.heappop(priority_queue)

    allocation[person] = allocation.get(person, 0) + 1

    vaccine_doses -= 1


# Display the allocation

print("Vaccine allocation:")

for person, doses in allocation.items():

    print(f"{person}: {doses} doses
```

VACCINE PRODUCTION

Creating a full-fledged Python code for vaccine production involves complex processes related to pharmaceutical manufacturing, which cannot be represented in a simple code snippet. Vaccine production involves multiple steps such as cell culture, purification, formulation, filling, quality control, and more.

PROGRAM

```
()class VaccineProduction:
   def _init_(self, name, production_capacity):
      self.name = name
      self.production_capacity = production_capacity
      self.current_inventory = 0

   def produce_vaccine(self, quantity):
      # Simulate vaccine production
      if self.current_inventory + quantity <= self.production_capacity:
         self.current_inventory += quantity
         print(f"Produced {quantity} doses of {self.name} vaccine.")
      else:
         print("Production capacity exceeded. Cannot produce more doses.")

   def check_inventory(self):
      print(f"Current inventory of {self.name} vaccine: {self.current_inventory}
doses.")

# Example usage
if _name_ == "_main_":
   pfizer_vaccine = VaccineProduction("Pfizer", production_capacity=10000)
   moderna_vaccine = VaccineProduction("Moderna",
production_capacity=8000)

   pfizer_vaccine.produce_vaccine(5000)
   pfizer_vaccine.check_inventory()


   moderna_vaccine.produce_vaccine(9000)
   moderna_vaccine.check_inventory
```

FEEDBACK COLLECTION

Creating a Python code for feedback collection typically involves defining functions or methods to gather feedback from users. Below is a simple example using a function to collect feedback:

PROGRAM

```
def collect_feedback():
    print("Thank you for using our service. Please provide your feedback below:")
    feedback = input("Feedback: ")

    # Save the feedback to a file or database
    with open("feedback.txt", "a") as file:
        file.write(feedback + "\n")

    print("Thank you for your feedback!")

# Example usage
if _name_ == "_main_":
    collect_feedback()
```