Task: Create a set example using these inbuilt methods

- clear()
- pop()

```
nums = {1, 2, 3, 4}
x = nums.pop()
print('Element popped : ' , x)
print(nums)
nums.clear()
print(nums)
```

```
Element popped :  1
{2, 3, 4}
set()
```

1. Create a function called execution_pipeline that processes a data pipeline using only positional arguments function signature must not change

- no keyword arguments allowed while calling
- function must rely strictly on argument order ex: def execute_pipeline(source, transform, validate, *steps):

```
def execution_pipeline(source, transform, validate):
  x = len(source)
  trans_msg = trans[:x]
  if source == trans_msg:
    validate = True
  return validate

src = input("Enter your source message : ")
trans = input("Enter your transformed message : ")
val = False
if execution_pipeline(src, trans, val):
  print('valid')
else:
  print('invalid')
```

```
Enter your source message : abcd
Enter your transformed message : abcd01
valid
```

2.Design a function named deploy_service that accepts only keyword arguments

and configures a deployment pipeline

- function must fail if any positional argument is passed
- all inputs must be passed as key=value

```
def deploy_service(*, service_name, available):
  if available:
    print(f"{service_name} is deployed successfully")
  else:
    print(f"{service_name} is not available")

sname = input('Enter service name:')
avail = input('Enter 1 if available and 0 if not available: ')
if avail not in ('0', '1'):
    raise ValueError('Check your input')
avail = bool(int(avail))

deploy_service(service_name = sname, available = avail)
```

```
Enter service name:aws
Enter 1 if available and 0 if not available: 1
aws is deployed successfully
```

3. create a function called create_user_profile that builds a user profile using default arguments correctly

- ex: def create_user_profile (username, role="user", active=True, skills=None): ..... username must be non empty string skills must be a list return distionary with profile details

```
def create_user_profile(username, role="user", active=True, skills=None):
  return {"username": username, "role": role, "active": active, "skills": skills}

name = input("Enter your username : ")
if len(name)<2:
    print("Username is invalid")
role = input("Enter your role : ")
active = input("Enter 1 for your active status and if not 0: ")
if active not in ('0', '1'):
    raise ValueError('Check your input')
active = bool(int(active))
skills = list(input("Enter your skills: ").split(','))

print('Profile: ',create_user_profile(name, role, active, skills))
```

```
Enter your username : yash
Enter your role : SD
Enter 1 for your active status and if not 0: 1
Enter your skills: python, java
Profile:  {'username': 'yash', 'role': 'SD', 'active': True, 'skills': ['python', ' java']}
```

4. Calling inner function from a nested function

```
def func1():
  a = 'sushma'

  def func2():
    print(a)

  return func2

# driver code
f2 = func1()
f2()
```

```
sushma
```

5. Tasks using lambda function

- task 1 : create a code using map() function under lambda condition
- task 2 : create a code using reduce() function under lambda condition from functools import reduce()

```
from functools import reduce
a = [1, 2, 3, 4]
b = list(map(lambda x: x+2, a))
print(b)
c = reduce(lambda x, y: x*y, a)
print(c)
```

```
[3, 4, 5, 6]
24
```

6. Write a function to check a number is prime or not.

```
def prime(num):
  count = 0
  if num < 2:
    return False
  for i in range(2, num//2 + 1):
    if num % i == 0:
      count = 1
      break
  if count == 0 :
    return True
  else:
    return False

inp = int(input("Enter a number : "))
if prime(inp):
  print("Prime")
else:
  print("Not Prime")
```

```
Enter a number : 57
Not Prime
```