```
# inheritance

class Phone: #base class / parent class
  def __init__(self, brand,model_name, price):
    self.brand = brand
    self.model_name = model_name
    self._price = price


  def full_name(self):
    return f"{self.brand} {self.model_name}"

  def make_a_call(self,number):
    return f"calling {number}...."

class Smartphone(Phone):
  def __init__(self, brand,model_name, price, ram, internal_memory, rear_camera):
    #two ways
    Phone.__init__(self,brand,model_name, price)
    self.ram = ram
    self.internal_memory = internal_memory
    self.rear_camera = rear_camera



phone = Phone('nokia', '1100','1000')
smartphone = Smartphone('oneplus','5',30000, '6 GB', '64 GB', '20 mp')
print(phone.full_name())
```



```
nokia 1100
oneplus 5
```

```
# Define greatest

def greatest(a,b,c):
  if a>b and a>c:
    return a
  elif b>a and b>c:
    return b
  else:
    return c

print(greatest(10,40,30))
```

```
40
```

```
# Define is_palindrome function that take one word in string as input
# and return True if it is palindrome else return False

# palindrome - word that reads same backwards as forwards
# example

# is_palindrome("madam") -----> True
# is_palindrome("naman") -----> True
# is_palindrome("horse") -----> False

# logic (alogrithm)
# step 1 -> reverse the string
# step 2 -> compare reversed string with original string


# solution

def is_palindrome(word):
```

```
    reversed_word = word[::-1]
    if word == reversed_word:
      return True
    else:
      return False

print(is_palindrome("naman"))
print(is_palindrome("horse"))
```

```
True
False
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

✓ 0s completed at 8:17 PM