```
# answer 1 - here def creates a function named add_item and passes two different arguments , item and an empty list box []
#            python creates the default list only once and reuses it since list is mutable thus the list persists between calls
#            def add_item(item , box =()) , here we created a tuple which is non mutable so must be created everytime

# answer 2 - difference is the way they return , __str__ is user friendly while __repr__ is for developer end .
#            if str is not there then repr is used as a fall back

# answer 3 - variable defined directly under the class name is createad only once for all objects while when defined inside
#            the init the variable is defined individually for each object and it might have different value for different obj.

#            if class variable is changed it is change for every time it is going to be used in fututre , while changing the
#            instance variable will only change the value for that object .
```

```
#ANSWER 4
# inp = input("log string : ")
# status = {}
# event = inp.split(';')
# for entry in event:
#     entry = entry.strip()
#     if entry == "":
#      continue
#     user, action = entry.split(':')
#     user = user.strip()
#     action = action.strip()
#     if user not in status:
#      status[user] = "Offline"
#     if action == "Login":
#         status[user] = "Online"
#     elif action == "Logout":
#         status[user] = "Offline"
# print (status)
```

```
# ANSWER 5
# while 1:
#  x = input ("sum number : ")
#  y = input ("second number : ")
#  z = input ("operation : ")
#  if x.isdigit() and y.isdigit():
#   if z == '+':
#      result = float(x)+float(y)
#   elif z == '-':
#      result = float (x)-float (y)
#   elif z == '*':
#      result = float (x) * float (y)
#   elif z == '/':
#      if y == '0':
#          print("Cannot divide by zero")
#          run=1
#      else :
#          result = float (x)/float (y)
#   else :
#      print ("invalide operator")
#      run =1
#  else :
#     print ( "ValueError")
#     run =1
#  if run !=1 :
#     print (result)
#  print ("Execution attempt complete")
```

```
#ANSWER 6
# class book :
#     def __init__(self , title , author ):
#         self.title = title
#         self.author = author
#         self.is_checked_out = False
# class library :
#     def __init__(self):
#         self.books = []
#     def add_book(self,book_obj):
#         self.books.append(book_obj)
#     def checkout_book(self,title):
#         for book in self.books:
#          if book.title == title:
#             if book.is_checked_out:
#                 print ("already checked out")
#             else :
#                 book.is_checked_out = True
#     def return_book (self,title):
#          for book in self.books:
#            if book.title == title:
#             book.is_checked_out = False
#     def book_status (self):
#         for book in self.books:
#            if book.is_checked_out :
#               print ("not available")
#            else :
#               print ("Available")
```

```
#ANSWER 7
# class Employee :
#   def __init__(self,sum ,last,salary):
#      self.sum = sum
#      self.last = last
#      self.salary = salary
#   def email(self):
#       return f"{self.first}.{self.last}@company.com"
#   def salary_setter (self,salary):
#       if salary <0:
#              print ("ERROR")
#   def fullname_deleter (self):
#       self.first = None
#       self.last = None

#ANSWER 8
# class TimeDuration:
#     def __init__(self, hours, minutes):
#         total = hours * 60 + minutes
#         self.hours = total // 60
#         self.minutes = total % 60

#     def __add__(self, other):
#         sum_hours = self.hours + other.hours
#         sum_minutes = self.minutes + other.minutes
#         return TimeDuration(sum_hours, sum_minutes)

#     def __str__(self):
#         return f"{self.hours}H:{self.minutes}M"
```