./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1 | 10/12/20 | Rithesh R Prabhu |  |  | Code without oops |
| 2 | 11/12/20 | Rithesh R prabhu |  |  | Code with oops |
| 3 | 12/12/20 | Rithesh R Prabhu |  |  | Code with oops and RE |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Table of Contents

[Table of Contents 3](#_Toc58819751)

[1: Python Code without OOPs 4](#_Toc58819752)

[1.1: Source code 4](#_Toc58819753)

[2: Python Code with OOPS 7](#_Toc58819754)

[2.1: Source code 7](#_Toc58819755)

List of Figures:

[Figure 1:Code Quality 6](#_Toc58819790)

[Figure 2: output of the project 6](#_Toc58819791)

[Figure 3:Code Quality (oops concept) 10](#_Toc58819792)

[Figure 4:ouput(oops concept) 10](#_Toc58819793)

## 1: Python Code without OOPs

### 1.1: Source code

# importing date and time model

import datetime

# Enter the time in HH:MM format 12 hour clock

print("Enter the time in 12 hours format(HH:MM) ")

alarm\_hour = int(input("Enter HH: "))

# The hour must be between 1 and 12

if (alarm\_hour > 13 or alarm\_hour < 1):

print("please enter a valid time")

exit()

alarm\_minute = int(input("Enter MM: "))

# The minute must be between 0 and 60

if (alarm\_minute > 61 or alarm\_minute < 0):

print("please enter a valid time")

exit()

# The time convention is am or pm

alarm\_am\_pm = str(input("am or pm: "))

if (alarm\_am\_pm != "am" and alarm\_am\_pm != "pm"):

print("please enter valid details")

exit()

if (alarm\_am\_pm == 'pm'):

alarm\_hour = alarm\_hour+12

# Creating a function for functioning of the clock,

# checks the time and hour

def alarm\_clock():

while(1):

if (alarm\_hour == datetime.datetime.now().hour and

alarm\_minute == datetime.datetime.now().minute):

print("wake up now")

break

# calling alarm\_call function

alarm\_clock()

# defining a function named snooze

def alarm\_snooze():

global alarm\_minute, alarm\_hour, alarm\_am\_pm

alarm\_minute1 = alarm\_minute

alarm\_hour1 = alarm\_hour

miss\_var1 = str(input("Enter Y to snooze "))

if (miss\_var1 == "Y"):

alarm\_snooze\_min = int(input("Enter the snooze time in MM format: "))

if (alarm\_snooze\_min > 31 or alarm\_snooze\_min < 0):

print("Enter a valid time: ")

exit()

alarm\_minute1 = alarm\_minute1+alarm\_snooze\_min

if(alarm\_minute1 > 61):

alarm\_hour1 = alarmhour1+1

if (alarm\_hour1 > 12):

alarm\_hour1 = 1

alarm\_minute1 = alarm\_minute1+alarm\_snooze\_min-60

while(1):

if (alarm\_hour1 == datetime.datetime.now().hour and

alarm\_minute1 == datetime.datetime.now().minute):

print("wake up now")

break

# calling a function

alarm\_snooze()

Github link: <https://github.com/99003188/pythonminiproject>

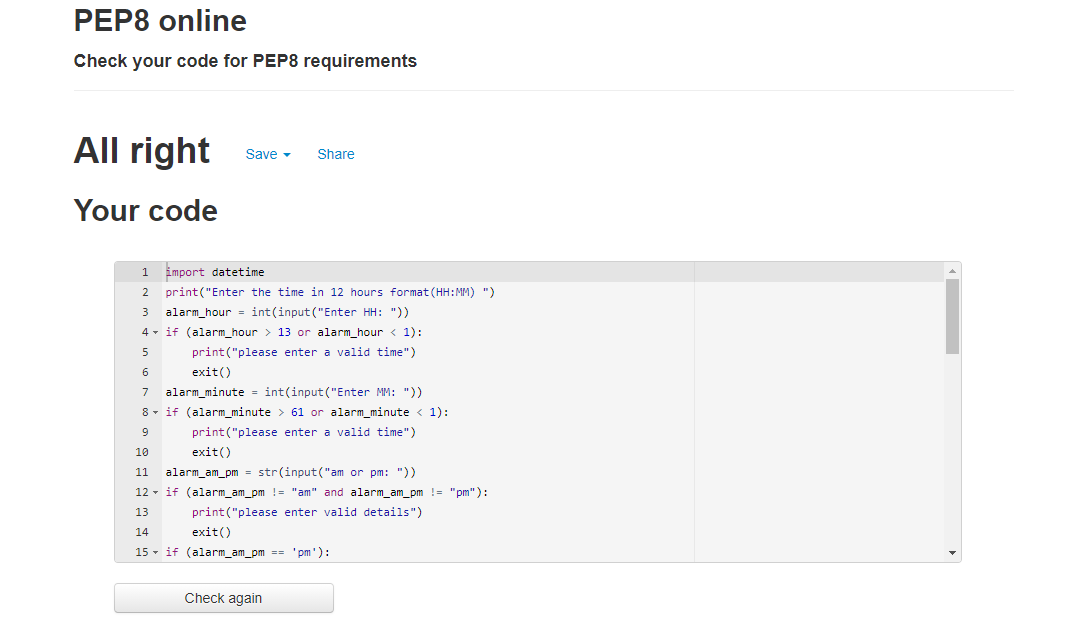


Figure :Code Quality

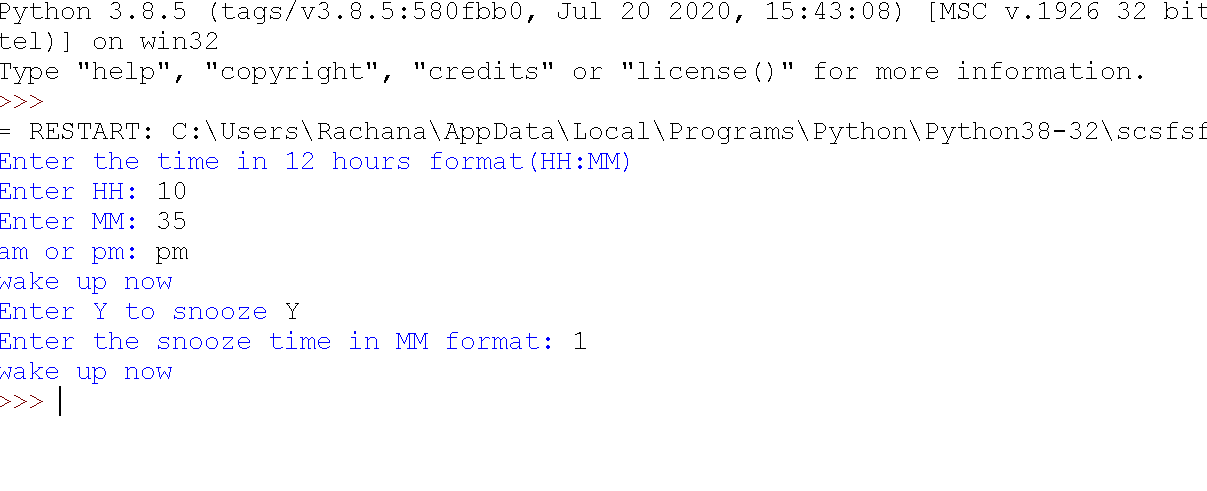


Figure : output of the project

## 

## 2: Python Code with OOPS

### 2.1: Source code

# import datetime module

import datetime

# import regular expression module

import re

# class for main setup alarm

class alarm():

print("Enter the time in 12 hours format(HH:MM) ")

# constructor to intialize variables required for function

def \_\_init\_\_(self, alarm\_minute=0, alarm\_hour=0, alarm\_am\_pm=0):

self.alarm\_hour = int(input("Enter HH: "))

self.alarm\_minute = int(input("Enter MM: "))

self.alarm\_am\_pm = str(input("am or pm: "))

self.lg = self.label()

# method for setup for alarm

def alarm\_setup(self):

if (self.alarm\_hour > 13 or self.alarm\_hour < 1):

print("please enter a valid time")

exit()

if (self.alarm\_minute > 61 or self.alarm\_minute < 0):

print("please enter a valid time")

exit()

if (self.alarm\_am\_pm != "am" and self.alarm\_am\_pm != "pm"):

print("please enter valid details")

exit()

if (self.alarm\_am\_pm == 'pm'):

self.alarm\_hour = self.alarm\_hour+12

while(1):

if (self.alarm\_hour == datetime.datetime.now().hour and

self.alarm\_minute == datetime.datetime.now().minute):

print("wake up now")

break

# creating a class for label, which is used to note down important requirements

class label:

def label\_show(self):

enter\_label = str(input("Enter the label"))

# creating a class for snooze option

class snooze(alarm):

# creating a constructor for intializing the variables

def \_\_init(self, alarm\_minute1=0, alarm\_hour1=0):

self.alarm\_minute1 = self.alarm\_minute

# creating a method called as snooze

def alarm\_snooze(self):

miss\_var1 = str(input("Enter Y to snooze "))

if (miss\_var1 == "Y"):

alarm\_snooze\_min = int(input("Enter the snooze time in MM format"))

else:

exit()

self.alarm\_minute = self.alarm\_minute + alarm\_snooze\_min

if (alarm\_snooze\_min > 31 or alarm\_snooze\_min < 0):

print("Enter a valid time: ")

exit()

if(self.alarm\_minute > 61):

self.alarm\_hour = self.alarmhour+1

if (self.alarm\_hour > 12):

self.alarm\_hour = 1

self.alarm\_minute = self.alarm\_minute+alarm\_snooze\_min-60

# method to print after alarm

def alarm\_snooze\_function(self):

if (self.alarm\_am\_pm == 'pm'):

self.alarm\_hour = self.alarm\_hour+12

while(1):

if (self.alarm\_hour == datetime.datetime.now().hour and

self.alarm\_minute == datetime.datetime.now().minute):

print("wake up now")

break

# creating a class to display the final comments

class display(snooze):

def display\_output(self):

print("Thank you for using alarm clock")

object1 = alarm()

object2 = snooze()

object3 = object2.lg

object4 = display()

object1.alarm\_setup()

object3.label\_show()

object2.alarm\_snooze()

object2.alarm\_snooze\_function()

object4.display\_output()

# regular expressions

print(re.search("[a-z]+[1-9][@][a-z]+[.][a-z]+", "prabhurithesh1@gmail.com"))

Github link: <https://github.com/99003188/pythonminiproject>



Figure :Code Quality (oops concept)

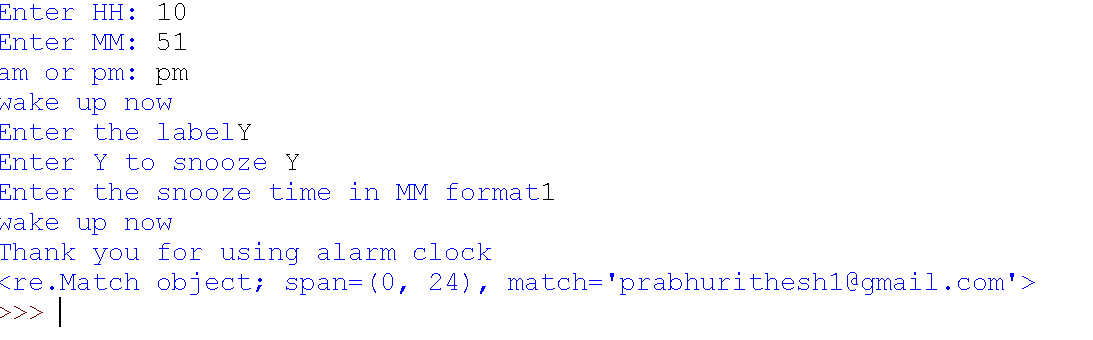


Figure :ouput(oops concept)