./

GENESIS – Advanced Python Programming



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1.0 | 11/12/2020 | Lakshmi N |  | Srinivas K |  |
| 1.1 | 12/12/2020 | Lakshmi N |  | Srinivas K |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

# **CURRENCY CONVERTER**

**Table of Contents**

[ACTIVITY 1 4](#_Toc58750807)

[1 PYTHON CODE 4](#_Toc58750808)

[2 PEP8 SCREENSHOT 8](#_Toc58750809)

[3 INPUT FILE SCREENSHOT 8](#_Toc58750810)

[4 OUTPUT FILE SCREENSHOT 9](#_Toc58750811)

[ACTIVITY 2 10](#_Toc58750812)

[5 OOPS CONCEPT 10](#_Toc58750813)

[6 PEP8 SCREENSHOT 15](#_Toc58750814)

[7 REGEX PRACTICE 16](#_Toc58750815)

[**7.1** **CODE:** 16](#_Toc58750816)

[**7.2** **OUTPUT:** 17](#_Toc58750817)

**List of Figures**

[Figure 1: PEP8 SCREENSHOT 8](file:///C:\Users\HP\Downloads\GENESIS%20-%20Advanced%20Python%20Programming.docx#_Toc58750822)

[Figure 2: INPUT FILE SCREENSHOT 8](file:///C:\Users\HP\Downloads\GENESIS%20-%20Advanced%20Python%20Programming.docx#_Toc58750823)

[Figure 3: OUTPUT FILE SCREENSHOT 9](file:///C:\Users\HP\Downloads\GENESIS%20-%20Advanced%20Python%20Programming.docx#_Toc58750824)

[Figure 4: PEP8 SCREENSHOT-OOPS CONCEPT 15](file:///C:\Users\HP\Downloads\GENESIS%20-%20Advanced%20Python%20Programming.docx#_Toc58750825)

[Figure 5: OUTPUT OF REGEX CODE 17](file:///C:\Users\HP\Downloads\GENESIS%20-%20Advanced%20Python%20Programming.docx#_Toc58750826)

# ACTIVITY 1

# PYTHON CODE

# Currency Converter

# Value of 1 USD = 76.65 INR

usd = 76.65

# Value of 1 EURO = 89.52 INR

euro = 89.52

# Set to 1 when file is being written

file\_handling\_flag = 0

# USD to INR conversion

class usd\_inr:

USD = 0.0

def conversion(self, amount):

USD = amount\*usd

if(file\_handling\_flag == 0):

print("The amount in INR is: ", "{0:.2f}".format(USD))

else:

test\_obj1 = input\_output\_file()

test\_obj1. testing\_write("USD", "INR", amount,

"{0:.2f}".format(USD))

# INR to USD conversion

class inr\_usd:

INR = 0.0

def conversion(self, amount):

INR = amount/usd

# print("The amount in USD is: ",INR)

if(file\_handling\_flag == 0):

print("The amount in USD is: ", "{0:.2f}".format(INR))

else:

test\_obj1 = input\_output\_file()

test\_obj1.testing\_write("INR", "USD", amount,

"{0:.2f}".format(INR))

# EURO to INR conversion

class euro\_inr:

EURO = 0.0

def conversion(self, amount):

EURO = amount\*euro

# print("The amount in INR is: ",EURO)

if(file\_handling\_flag == 0):

print("The amount in INR is: ", "{0:.2f}".format(EURO))

else:

test\_obj1 = input\_output\_file()

test\_obj1.testing\_write("EURO", "INR", amount,

"{0:.2f}".format(EURO))

# INR to EURO conversion

class inr\_euro:

INR = 0.0

def conversion(self, amount):

INR = amount/euro

# print("The amount in EURO is: ",INR)

if(file\_handling\_flag == 0):

print("The amount in EURO is: ", "{0:.2f}".format(INR))

else:

test\_obj1 = input\_output\_file()

test\_obj1.testing\_write("INR", "EURO", amount,

"{0:.2f}".format(INR))

# File Handling for testing

class input\_output\_file:

def testing\_read(self):

fr = open("inputfile.txt", "r")

for line in fr:

option = int(line[0])

value = ''

for i in range(2, len(line)):

value = value + line[i]

value = float(value)

menu(option, value)

fr.close()

def testing\_write(self, curr1, curr2, given\_value, conv\_value):

fw = open("outputfile.txt", "a")

output\_str = str(str(given\_value) + " " + curr1 + " --> " +

curr2 + " " + str(conv\_value) + "\n")

fw.write(output\_str)

fw.close()

# Menu function

def menu(x, amount):

if (x == 1):

obj = usd\_inr()

obj.conversion(amount)

elif (x == 2):

obj = inr\_usd()

obj.conversion(amount)

elif (x == 3):

obj = euro\_inr()

obj.conversion(amount)

elif (x == 4):

obj = inr\_euro()

obj.conversion(amount)

elif (x > 5):

print("Invalid choice !!!")

print(" \t\t\tCURRENCY CONVERTER \n\n")

print("Options:")

print("\t1. USD ---> INR")

print("\t2. INR ---> USD")

print("\t3. EURO ---> INR")

print("\t4. INR ---> EURO")

print("\t5. Testing file handling")

value = int(input("Enter your choice "))

if(value < 5):

amount = float(input("Enter amount to be converted: "))

menu(value, amount)

if(value == 5):

test\_obj = input\_output\_file()

file\_handling\_flag = 1

test\_obj.testing\_read()

print("\noutputfile.txt updated using inputs from inputfile.txt\n")

# PEP8 SCREENSHOT

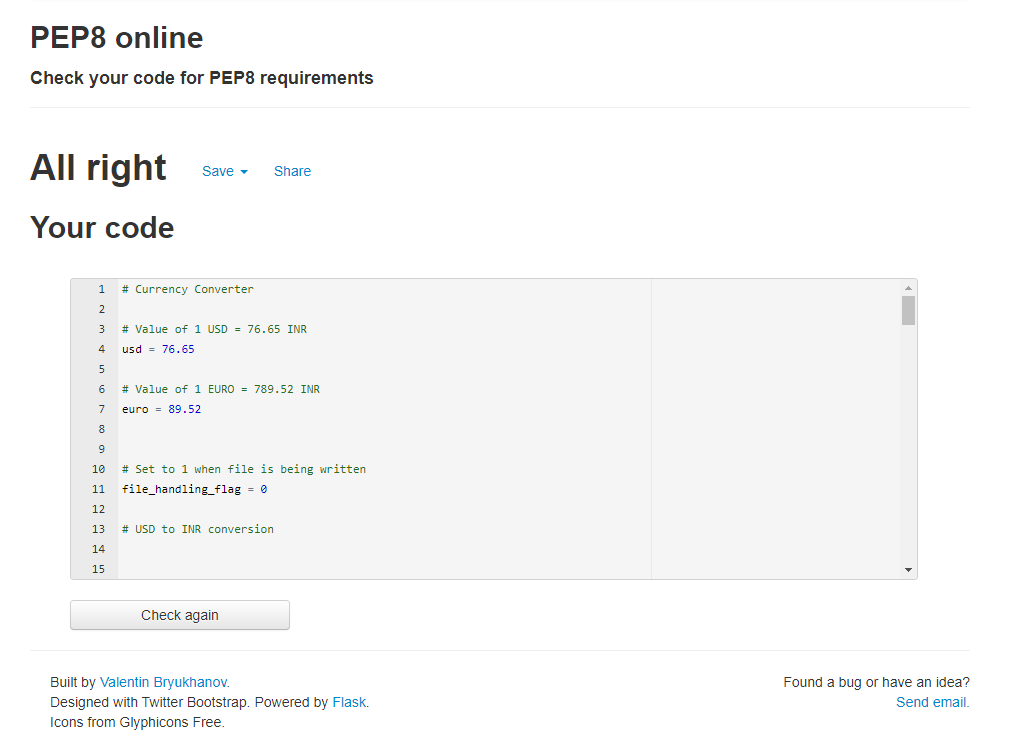


Figure : PEP8 SCREENSHOT

# INPUT FILE SCREENSHOT

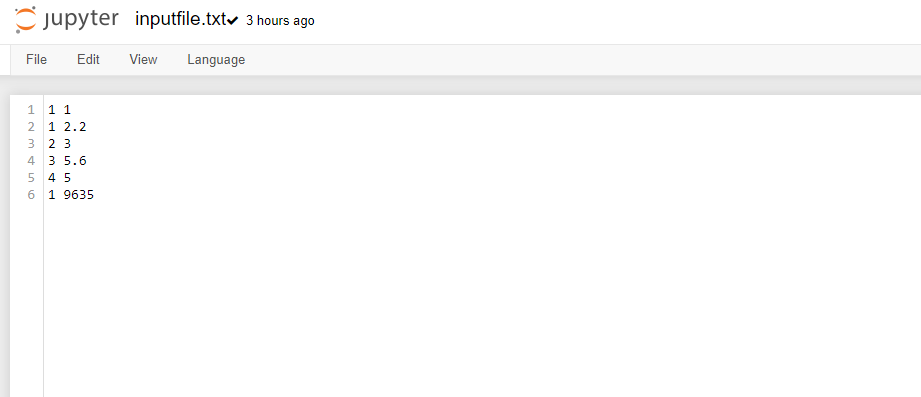
* Enter the choice as 5 to check the file handling.

Figure : INPUT FILE SCREENSHOT

# OUTPUT FILE SCREENSHOT

Figure : OUTPUT FILE SCREENSHOT

# ACTIVITY 2

# OOPS CONCEPT

# Currency Converter

import requests

url = str.\_\_add\_\_('http://data.fixer.io/api/latest?access\_key=',

'5934bf3f2adef071c95f87d808b3bb28')

class CurrencyConverter:

# empty dict to store the conversion rates

rates = {}

def \_\_init\_\_(self, url):

data = requests.get(url).json()

self.rates = data["rates"]

def convert(self, from\_currency, to\_currency, amount):

if from\_currency != 'EUR':

amount = amount / self.rates[from\_currency]

# limiting the precision to 2 decimal places

amount = round(amount \* self.rates[to\_currency], 2)

return amount

# Set to 1 when file is being written

file\_handling\_flag = 0

# USD to INR conversion

class usd\_inr(CurrencyConverter):

def \_\_init\_\_(self, x):

super().\_\_init\_\_(url)

self.x = self.convert("USD", "INR", 1)

def conversion(self, amount):

try:

assert amount > 0

USD = amount \* self.x

if(file\_handling\_flag == 0):

print("The amount in INR is: ", "{0:.2f}".format(USD))

else:

test\_obj1 = input\_output\_file()

test\_obj1. testing\_write("USD", "INR", amount,

"{0:.2f}".format(USD))

except AssertionError:

print("Invalid input")

# INR to USD conversion

class inr\_usd(usd\_inr):

def conversion(self, amount):

try:

assert amount > 0

INR = amount/self.x

if(file\_handling\_flag == 0):

print("The amount in USD is: ", "{0:.2f}".format(INR))

else:

test\_obj1 = input\_output\_file()

test\_obj1.testing\_write("INR", "USD", amount,

"{0:.2f}".format(INR))

except AssertionError:

print("Invalid input")

# EURO to INR conversion

class euro\_inr(CurrencyConverter):

def \_\_init\_\_(self, x):

super().\_\_init\_\_(url)

self.x = self.convert("EUR", "INR", 1)

def conversion(self, amount):

try:

assert amount > 0

EURO = amount\*self.x

if(file\_handling\_flag == 0):

print("The amount in INR is: ", "{0:.2f}".format(EURO))

else:

test\_obj1 = input\_output\_file()

test\_obj1.testing\_write("EURO", "INR", amount,

"{0:.2f}".format(EURO))

except AssertionError:

print("Invalid input")

# INR to EURO conversion

class inr\_euro(euro\_inr):

def conversion(self, amount):

try:

assert amount > 0

INR = amount/self.x

if(file\_handling\_flag == 0):

print("The amount in EURO is: ", "{0:.2f}".format(INR))

else:

test\_obj1 = input\_output\_file()

test\_obj1.testing\_write("INR", "EURO", amount,

"{0:.2f}".format(INR))

except AssertionError:

print("Invalid input")

# EURO to USD conversion

class euro\_usd(CurrencyConverter):

def \_\_init\_\_(self, x):

super().\_\_init\_\_(url)

self.x = self.convert("EUR", "USD", 1)

def conversion(self, amount):

try:

assert amount > 0

EURO = amount\*self.x

if(file\_handling\_flag == 0):

print("The amount in USD is: ", "{0:.2f}".format(EURO))

else:

test\_obj1 = input\_output\_file()

test\_obj1.testing\_write("EURO", "USD", amount,

"{0:.2f}".format(EURO))

except AssertionError:

print("Invalid input")

# USD to EURO conversion

class usd\_euro(euro\_usd):

def conversion(self, amount):

try:

assert amount > 0

USD = amount/self.x

if(file\_handling\_flag == 0):

print("The amount in EURO is: ", "{0:.2f}".format(USD))

else:

test\_obj1 = input\_output\_file()

test\_obj1.testing\_write("USD", "EURO", amount,

"{0:.2f}".format(USD))

except AssertionError:

print("Invalid input")

# File Handling for testing

class input\_output\_file:

def testing\_read(self):

fr = open("inputfile.txt", "r")

for line in fr:

option = int(line[0])

value = ''

for i in range(2, len(line)):

value = value + line[i]

value = float(value)

menu(option, value, url)

fr.close()

def testing\_write(self, curr1, curr2, given\_value, conv\_value):

fw = open("outputfile.txt", "a")

output\_str = str(str(given\_value) + " " + curr1 + " --> " +

curr2 + " " + str(conv\_value) + "\n")

fw.write(output\_str)

fw.close()

# Menu function

def menu(x, amount, url):

if (x == 1):

obj = usd\_inr(url)

obj.conversion(amount)

elif (x == 2):

obj = inr\_usd(x)

obj.conversion(amount)

elif (x == 3):

obj = euro\_inr(url)

obj.conversion(amount)

elif (x == 4):

obj = inr\_euro(x)

obj.conversion(amount)

elif (x == 5):

obj = euro\_usd(url)

obj.conversion(amount)

elif (x == 6):

obj = usd\_euro(url)

obj.conversion(amount)

elif (x > 8):

print("Invalid choice !!!")

print(" \t\t\tCURRENCY CONVERTER \n\n")

print("Options:")

print("\t1. USD ---> INR")

print("\t2. INR ---> USD")

print("\t3. EURO ---> INR")

print("\t4. INR ---> EURO")

print("\t5. EURO ---> USD")

print("\t6. USD ---> EURO")

print("\t7. Testing file handling")

value = int(input("Enter your choice "))

if(value < 7):

amount = float(input("Enter amount to be converted: "))

menu(value, amount, url)

if(value == 7):

test\_obj = input\_output\_file()

file\_handling\_flag = 1

test\_obj.testing\_read()

print("\noutputfile.txt updated using inputs from inputfile.txt\n")

# PEP8 SCREENSHOT

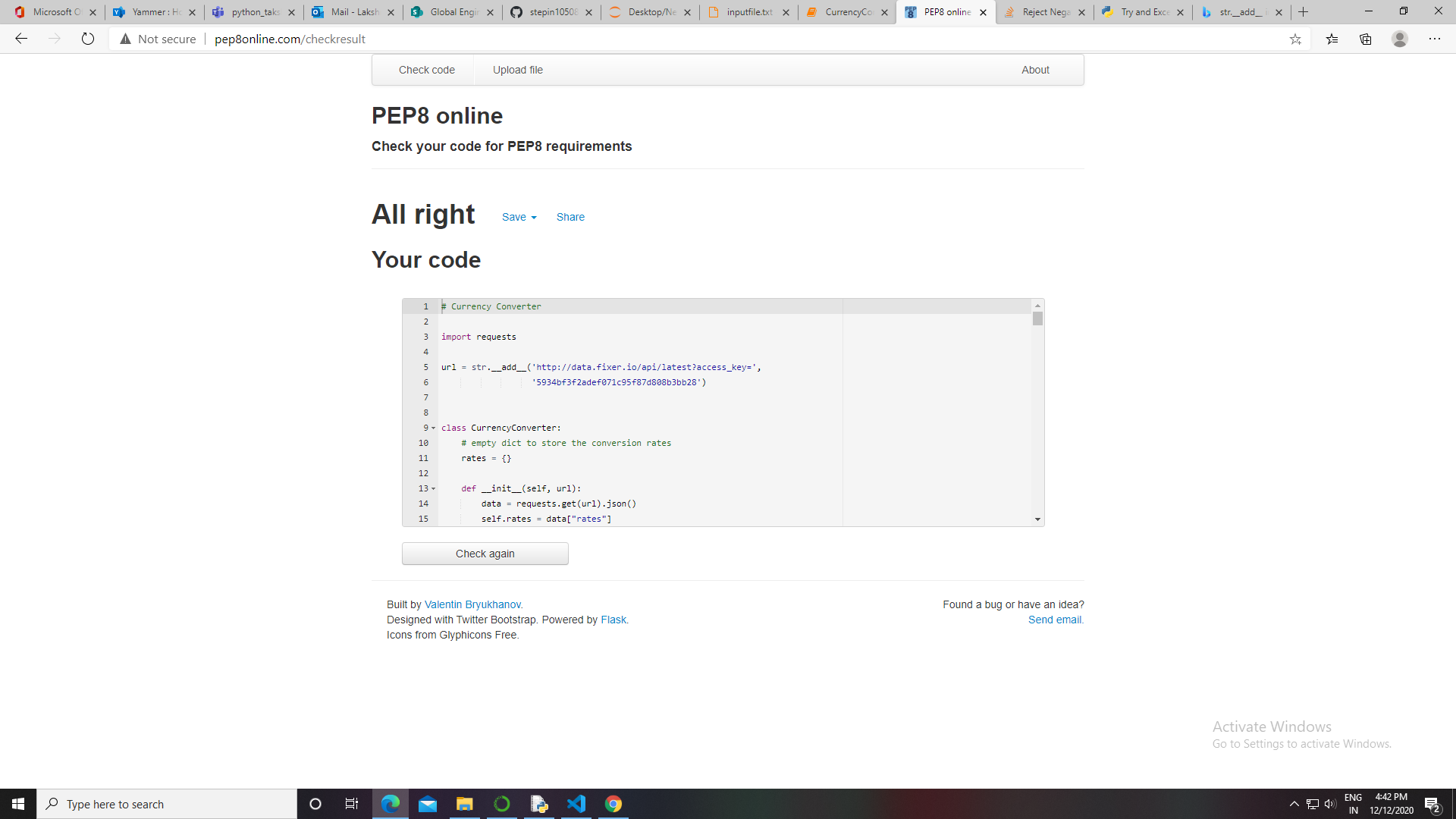


Figure : PEP8 SCREENSHOT-OOPS CONCEPT

# REGEX PRACTICE

## **CODE:**

import re

sample = '''SCIENTISTS HAVE LEARNED TO SUPPLEMENT THE SENSE OF SIGHT IN NUMEROUS WAYS.

In front of the tiny pupil of the eye they put, on Mount Palomar, a great monocle 200 inches+ in diameter

and with it see 2000 times farther into the depths of space.

Or they look through a small pair of lenses arranged as a microscope\* into a drop of water or blood,

and magnify by as much as 2000 diameters the living creatures there,

many of which are among man’s most dangerous enemies.'''

print(re.match("[A-Z a-z]{0,10}",sample))

print(re.match("[a-z A-Z]", sample))

print(re.match("[0-9 a-z]+", sample))

print(re.search("[0-9]", sample))

print(re.match("[0-9]{2,3}", sample))

print(re.findall("[0-9]{2,3}", sample))

print(re.findall("[+]", sample))

print(re.match("^[A-Z a-z]?", sample))

print(re.match("[A-Z a-z]?", sample))

print(re.match("[a-z]\*", sample))

print(re.findall("ti?mes", sample))

print(re.match("\D\S", sample))

print(re.match("SCIE?NTISTS", "SCINTISTS"))

print(re.findall("look?[a-z]+", sample))

print(re.match("[a-z]+\.+[a-z]+@+[a-z]+.+[a-z]", "lakshmi.n@ltts.com"))

print(re.match("[a-z]+\.+[a-z]+@+[a-z]+.+[a-z]", "lakshmin123@ltts.com"))

print(re.match("[a-z]+[0-9]+@+[a-z]+.+[a-z]", "lakshmin123@ltts.com"))

print(re.findall("enemies.\Z", sample))

## **OUTPUT:**

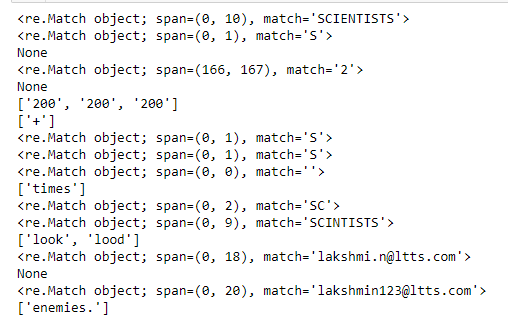


Figure : OUTPUT OF REGEX CODE