Main del programa:

```
import funciones as fcf

def main():
    textoS = fcf.lector(100,350)
    textoL = fcf.limpieza(textoS)
    lista_palabras = fcf.separa_texto(textoL)
    dict = fcf.diccionario_obj(lista_palabras)
    fcf.crea_excel(dict)
    tamanio, referencia, probabilidad = fcf.menu()
    textoFinal = fcf.crea_oracion(dict, tamanio, probabilidad,
referencia)
    textoFinal = fcf.mejora_oracion(textoFinal)
    fcf.escribe_bitacora("La oracion final es: " + textoFinal)
    print(textoFinal)

main()
```

Funciones del programa:

```
from PyPDF2 import PdfReader
from datetime import datetime
import re
from os import system
from palabra import *
import xlsxwriter as xl


def menu():
    tamanio     = input("¿Cuántas palabras desea que tenga la
oración?\n")
    referencia  = input("¿Dame la referencia para comenzar la
oración?\n")
    probabilidad= input("Deseas que la probabilidad de selección de
siguiente palabra sea:\n-A.Alta\n-B.Baja\n-M.Media\n")
    system("clear")
    return tamanio, referencia, probabilidad

def lector(inicio, fin):
    reader = PdfReader("Harry_Potter.pdf")
    for i in range(inicio,fin):
        page = reader.pages[i]
        pagina = page.extract_text()
        if i == inicio:
            libro = pagina
        else:
            libro = libro + " " + pagina
    escribe_bitacora("Se leyo el PDF y se extrajo el texto desde la
página " +  str(inicio) + " hasta " + str(fin) + "\n")
    return libro

def limpieza(text):
    basura = "—-_'!¡¿?…:;.,«»()"
    text = ''.join(x for x in text if x not in basura)
    text = re.sub("\n"," ",text)
    text = text.lower()
```

```
    #print(text)
    escribe_bitacora("Se realizo la limpieza del texto.\n")
    return text

def mejora_oracion(oracion):
    oracion = oracion.capitalize()
    #if oracion[:-1] == " ":
    oracion = oracion[:-1]
    oracion = oracion + "."

    escribe_bitacora("\nSe mejoro la oracion.\n")
    return oracion

def escribe_bitacora(registro):
    now = datetime.now()
    nombre_archivo = str(now.day) + '-' + str(now.month) + '-' +
str(now.year) + '_' +  str(now.hour) + 'horas_' + str(now.minute) +
'minutos_'
    file = open(nombre_archivo + '.txt', "a")
    file.write(registro)
    file.close()

def separa_texto(texto):
    return texto.split(" ")

def diccionario_obj(lista):
    diccionario = {}
    for i in range(len(lista)-1):
        if lista[i] not in diccionario.keys():
            #print("Guarda", palabra)
            diccionario[lista[i]] = Palabra(lista[i])
        else:
            diccionario[lista[i]].num_ocurrencia += 1

        diccionario[lista[i]].agrega_sig_palabra(lista[i+1])

    for i in diccionario.keys():
        diccionario[i].cierra_tabla()

    escribe_bitacora("Se crea un diccionario con todas las palabras del
texto y sus siguientes palabras.\n")
    return diccionario

def crea_oracion(diccionario, tamanio, probabilidad, referencia =
"harry"):
    textoF = ""
    for i in range(int(tamanio)):
        escribe_bitacora("Referencia: " +  referencia + "  sig palabras:
" + str(diccionario[referencia].tupla_sig_palabra) + "\n\n")
        textoF += referencia + " "
        aux = diccionario[referencia]
        referencia = aux.get_sig_palabra(probabilidad)
    return textoF

def crea_excel(diccionario):
    archivo = xl.Workbook('Base_datos.xlsx')
    hoja=archivo.add_worksheet()
    i = 0
    for llave, valor in diccionario.items():
        hoja.write(i,0, llave)
```

```
        hoja.write(i,1, valor.num_ocurrencia+1)
        hoja.write(i,2, str(valor.tupla_sig_palabra))
        i+=1
    archivo.close()
```

Clase palabra del programa

```
import operator
import random

class Palabra:

    def __init__(self, nombre):
        self.nombre = nombre
        self.num_ocurrencia    = 0
        self.num_sig_palabra   = 0
        self.diccionario_sig_palabra = {}
        self.tupla_sig_palabras = ()

    def agrega_sig_palabra(self, sig_palabra):
        if sig_palabra in self.diccionario_sig_palabra.keys():
            self.diccionario_sig_palabra[sig_palabra] += 1
        else:
            self.diccionario_sig_palabra[sig_palabra] = 1


    def elimina_sig_palabra(self, palabra_eliminar):
        aux = True
        if palabra_eliminar in self.diccionario_sig_palabra:
            del self.diccionario_sig_palabra[palabra_eliminar]
            self.conteo_a_probabilidad()
        else:
            aux = False
        return aux

    def conteo_a_probabilidad(self):
        aux = 0
        for conteo in self.diccionario_sig_palabra:
            aux = aux + self.diccionario_sig_palabra.get(conteo)
        for conteo in self.diccionario_sig_palabra:
            self.diccionario_sig_palabra[conteo] =
(self.diccionario_sig_palabra.get(conteo) /aux) * 100

    def cierra_tabla(self):
        self.conteo_a_probabilidad()
        self.ordena_diccionario()

    def get_sig_palabra(self, probabilidad):
        aux = len(self.tupla_sig_palabra)//3
        if probabilidad == 'a' or probabilidad == 'A':
            aleatorio = random.randint(0, aux)
            #return self.tupla_sig_palabra[aleatorio][0]
        elif probabilidad == 'b' or probabilidad == 'B':
            aleatorio =  len(self.tupla_sig_palabra) -1
            #aleatorio = random.randint(len(self.tupla_sig_palabra) - 2,
len(self.tupla_sig_palabra))
            #return
self.tupla_sig_palabra[len(self.tupla_sig_palabra)][0]
```

```
        elif probabilidad == 'm' or probabilidad == 'M':
            aleatorio = random.randint(aux, aux*2)
            #return
self.tupla_sig_palabra[len(self.tupla_sig_palabra)/2][0]
            #aleatorio = random.randint(len(self.tupla_sig_palabr)/2 - 1
, len(self.tupla_sig_palabra)/2+1)

        return self.tupla_sig_palabra[aleatorio][0]

#Para acceder a la clave se usa self.tupla_sig_palabra[1][0]
#Para acceder al valor se usa   self.tupla_sig_palabra[1][0]
    def ordena_diccionario(self):
        self.tupla_sig_palabra =
sorted(self.diccionario_sig_palabra.items(), key=operator.itemgetter(1),
reverse=True)
```