Guía de Ejercicios Prácticos - Ficha 10

Sitio: Universidad Virtual UTN FRC
Curso: Algoritmos y Estructuras de Datos (2020)
Libro: Guía de Ejercicios Prácticos - Ficha 10
Imprimido por: Luciana Lisette Montarce
Día: lunes, 23 de noviembre de 2020, 21:20

Descripción

Esta guía contiene enunciados de algunos ejercicios para aplicar los conceptos de programación en ***Python*** que se analizan en la ***Ficha 10***. Los alumnos no deben subir nada al aula virtual: la guía se propone como fuente de ejercicios generales. Se sugiere intentar resolver cada uno de estos problemas ya sea trabajando solos o en grupos de estudio, y cuando las soluciones se publiquen, controlar lo hecho con las sugerencias propuestas por sus docentes. Utilice el foro del curso para plantear dudas y consultas, que cualquier alumno puede intentar responder.

Tabla de contenidos

1. Sílaba "de" en la primera mitad

Desarrollar un programa en Python que permita cargar por teclado un texto completo. Siempre se supone que el usuario cargará un punto para indicar el final del texto, y que cada palabra de ese texto está separada de las demás por un espacio en blanco. El programa debe:

a) Determinar cuántas palabras tenían al menos un caracter que era en realidad un dígito (un caracter entre '0' y '9').

b) Determinar cuántas palabras tenían 3 o menos letras, cuántas tenían 4 y hasta 6 letras, y cuántas tenían más de 6 letras.

c) Determinar la longitud de la palabra más larga del texto.

d) Determinar cuántas palabras contuvieron la expresión "de", pero en la primera mitad de la palabra.

1.1. Solucion

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'


def is_digit(letra):
    digitos = '0123456789'
    return letra in digitos


def test():
    print('Analisis de Texto - Silaba \"de\" primera mitad')
    print('=' * 80)

    texto = input('Ingrese el texto a analizar, finaliza con punto: ')
    tiene_digito = tiene_d = tiene_de = False
    palabra_digito = pal_3letras = pal_4a6letras = pal_mas6letras = cant_letras = pal_demitad = 0
    mayor_longitud = posicion = 0

    for letra in texto:
        if letra == ' ' or letra == '.':

            if tiene_digito:
                palabra_digito += 1

            if cant_letras <= 3:
                pal_3letras += 1
            elif 4 <= cant_letras <= 6:
                pal_4a6letras += 1
            elif cant_letras > 6:
                pal_mas6letras += 1

            if cant_letras > mayor_longitud:
                mayor_longitud = cant_letras

            mitad = cant_letras // 2
            if tiene_de and 0 < posicion <= mitad:
                pal_demitad += 1

            tiene_de = tiene_d = False
            tiene_digito = False
            cant_letras = 0

        else:
            cant_letras += 1

            if is_digit(letra):
                tiene_digito = True

            if letra == 'd' and not tiene_de:
                tiene_d = True
            else:
                if letra == 'e' and tiene_d:
                    tiene_de = True
                    posicion = cant_letras
                tiene_d = False

    print('Presentacion de Resultados')
    print('-' * 80)
    print('La cantidad de palabras que contienen al menos un digito: ', palabra_digito)
    print('La cantidad de palabras que contienen hasta 3 letras es:', pal_3letras)
    print('La cantidad de palabras que contienen entre 4 y 6 letras es:', pal_4a6letras)
    print('La cantidad de palabras que contienen mas de 6 letras es:', pal_mas6letras)
    print('La cantidad de palabras con la expresion \"de\" en la primera mitad es:', pal_demitad)


test()
```

2. Sólo menores que 7.

Desarrollar un programa de Phyton que permita cargar por teclado un secuencia de números, uno por uno. Siempre se supone que el usuario cargará un 0(cero) para indicar el final del proceso de carga. El cero no debe considerarse un dato a procesar. El programa debe:

a) Determinar el porcentaje que cantidad de números pares representa en la cantidad total de números ingresados.

b) Determinar cuántos de los números ingresados tenían su último dígito igual a 4 o igual a 5.

c) Determinar el menor de los números ingresados que sean divisibles por 3.

d) Determinar si la secuencia estaba formada sólo por números menores o iguales que 7.

2.1. Solución

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'


def calcular_procentaje(cantidad, total):
    porcentaje = 0
    if total > 0:
        porcentaje = cantidad * 100 / total
    return porcentaje


def es_multiplo(numero, divisor):
    resto = numero % divisor
    return resto == 0


def es_par(numero):
    return es_multiplo(numero, 2)


def test():
    # variables para el punto 1
    cant_pares = 0
    cant_numeros = 0

    # variables para el punto 2
    cant_numeros_punto2 = 0

    # variables para el punto 3
    menor = None

    # variables para el punto 4
    # se inicializa con True, suponiendo que se ingresa una secuencia de numeros menores o iguales a 7
    # si dentro del ciclo se encuentra un numero que no cumple se cambia a false
    secuencia_menor_igual7 = True

    print('Ingrese una secuencia de números, la misma termina con un 0 (cero) : ')
    numero = int(input('Ingrese un número (distinto de 0): '))
    while numero != 0:
        cant_numeros += 1

        # punto 1
        # pregunta si es par
        if es_par(numero):
            cant_pares += 1

        # punto 2
        digito = numero
        if numero > 10:
            digito = numero % 10

        if digito == 4 or digito == 5:
            cant_numeros_punto2 += 1

        # punto 3
        # si el numero leido es divisible por 3 realiza proceso de busqueda de menor o menor no esta inicializado
        if es_multiplo(numero, 3):
            if menor is None or numero < menor:
                menor = numero

        # punto 4
        # si encuentra algun valor mayor a 7 ya no cumple
        if numero > 7:
            secuencia_menor_igual7 = False

        numero = int(input('Ingrese un número (distinto de 0): '))

    # punto 1
```

```
    # calcula el porcentaje
    porcentaje = calcular_procentaje(cant_pares, cant_numeros)

    # punto 4
    if secuencia_menor_igual7:
        mensaje = 'La secuencia estaba formada sólo por números menores o iguales que 7'
    else:
        mensaje = 'La secuencia NO estaba formada sólo por números menores o iguales que 7'

    # mostrar resultados
    print('\n El porcentaje de números pares sobre la cantidad total de números ingresados: ', porcentaje)
    print('\n La cantidad de números ingresados que tiene su último dígito igual a 4 o igual a 5: ', cant_numeros_punto2)
    print('\n El menor de los números ingresados que son divisibles por 3: ', menor)
    print('\n ', mensaje)


test()
```

3. Solamente 'a'

Desarrollar un programa que permita ingresar por teclado, con palabras separadas por un espacio y terminado en punto. En base al texto ingresado, determinar:

a) Cuál es la longitud de la palabra más larga.

b) Cuántas palabras tienen la a como única vocal

c) Qué porcentaje representan las que sólo tienen la vocal a sobre el total de palabras.

****

Ejemplo: "el agua clara salta por las piedras."

La longitud de la palabra más larga es 7 letras

Las palabras cuya única vocal es la a son: 3

El porcentaje de estas palabras sobre el total es 43 %

3.1. Solución

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'


def calcular_porcentaje(cantidad, total):
    if total == 0:
        porcentaje = 0
    else:
        porcentaje = round(cantidad * 100 / total)
    return porcentaje


def es_vocal(letra):
    vocales = 'aeiou'
    return letra in vocales


def test():
    print('Análisis de Texto')
    print('*' * 80)

    # Inicialización
    letras_pal = 0
    palabras = 0
    tiene_a = False

    tiene_eiou = False
    palabras_soloa = 0

    # Carga de datos y proceso
    texto = input('Ingrese el texto a analizar, separando las palabras con un espacio y terminando con punto: ')

    for letra in texto:
        if letra == ' ' or letra == '.':
            # Contar palabras
            if letras_pal > 0:
                palabras += 1

            # Buscar mayor
            if palabras == 1:
                mayor = letras_pal
            elif letras_pal > mayor:
                mayor = letras_pal
            # Tiene sólo a?

            if tiene_a and not tiene_eiou:
                palabras_soloa += 1

            # Reiniciar los indicadores de palabra
            letras_pal = 0
            tiene_a = False
            tiene_eiou = False
        else:
            # Determinar longitud
            letras_pal += 1

            # Detectar sólo a
            if es_vocal(letra):
                if letra == 'a':
                    tiene_a = True
                else:
                    tiene_eiou = True

    # Resultados
    print('*' * 80)
    print('La longitud de la palabra más larga es', mayor, 'letras')
    print('Las palabras cuya única vocal es la a son:', palabras_soloa)
    porcentaje = calcular_porcentaje(palabras_soloa, palabras)
```

```
    print('El porcentaje de estas palabras sobre el total es', porcentaje, '%')


# Script principal
test()
```

4. La letra T y los porcentajes

Desarrollar un programa en Python que permita cargar por teclado un texto completo. Siempre se supone que el usuario cargará un punto para indicar el final del texto, y que cada palabra de ese texto está separada de las demás por un espacio en blanco. El programa debe:

a) Determinar la cantidad de palabras en las que solo aparece una única vez la letra "t"

b) Determinar la cantidad de palabras cuya cantidad de letras es mayor a la cantidad de letras de la palabra anterior

c) Determinar la cantidad de palabras con la cantidad de letras pares y comienzan con la letra "c"

d) Determinar el porcentaje que representa el primer punto sobre el total de las palabras del texto procesado

4.1. Solución

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'


def calcular_porcentaje(cantidad, total):
    porc = 0
    if total >= 0:
        porc = cantidad * 100 / total
    return round(porc, 2)


def es_par(numero):
    resto = numero % 2
    return resto == 0


def test():
    cantidad_t = pal_unica_t = cant_letra = letras_pal_ant = cant_palabras = pal_mayor_anterior = pal_pares_con_c = 0
    empieza_c = False

    print('La Letra T - Procesamiento de Texto')
    print('=' * 80)

    texto = input('Ingrese el texto a procesar, finaliza con un punto:')

    for caracter in texto:
        if caracter != ' ' and caracter != '.':
            cant_letra += 1

            if caracter == 't':
                cantidad_t += 1

            if cant_letra == 1 and caracter == 'c':
                empieza_c = True

        else:
            if cant_letra > 0:

                cant_palabras += 1

                if cantidad_t == 1:
                    pal_unica_t += 1

                if cant_palabras == 1:
                    letras_pal_ant = cant_letra
                else:
                    if cant_letra > letras_pal_ant:
                        pal_mayor_anterior += 1
                    letras_pal_ant = cant_letra

                if empieza_c and es_par(cant_letra):
                    pal_pares_con_c += 1

                cant_letra = 0
                cantidad_t = 0
                empieza_c = False

    porcentaje = calcular_porcentaje(pal_unica_t, cant_palabras)

    print('Resultados')
    print('=' * 80)
    print('La cantidad de palabras con una sola t son:', pal_unica_t)
    print('y representan el ', porcentaje, '% del total de palabras del texto', sep='')
    print('La cantidad de palabras con cantidad pares de letras y empiezan con c son:', pal_pares_con_c)
    print('La cantidad de palabras que tienen mas letras que la anterior son:', pal_mayor_anterior)


test()
```

5. Con m y b

Desarrollar un programa que permita ingresar por teclado, con palabras separadas por un espacio y terminado en punto. En base al texto ingresado, determinar:

a) Cuántas palabras tienen una m y una b a partir de la tercer letra.

b) Cuántas palabras comienzan con la letra p seguida de cualquier vocal.

c) Cuántas palabras comienzan y terminan con el mismo carácter.

****

Ejemplo: 'Mi amiga Ambar siempre piensa y cambia pronto.'

****

Palabras tienen una m y una b a partir de la tercer letra: 1
Palabras que comienzan con la letra p seguida de cualquier vocal: 1
Palabras que comienzan y terminan con el mismo carácter: 2

5.1. Solución

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'


def es_vocal(letra):
    vocales = 'aeiou'
    return letra in vocales


def test():
    print('Análisis de Texto')
    print('*' * 80)

    # Inicialización
    letras_pal = palabras_bym = palabras_pvocal = palabras_iniciofin = 0
    tiene_b = tiene_m = tiene_p = tiene_pvocal = False
    primera = None
    ultima = None

    # Carga de datos y proceso
    texto = input('Ingrese el texto a analizar, separando las palabras con un espacio y terminando con punto: ')

    for letra in texto:
        if letra == ' ' or letra == '.':
            # Tiene m y b a partir de la tercer letra?
            if tiene_b and tiene_m:
                palabras_bym += 1

            # Empieza con p seguida de vocal?
            if tiene_pvocal:
                palabras_pvocal += 1

            # Empieza y termina con la misma letra?
            if primera == ultima:
                palabras_iniciofin += 1

            # Reiniciar los indicadores de palabra
            letras_pal = 0
            tiene_b = False
            tiene_m = False
            tiene_p = False
            tiene_pvocal = False
            primera = None
            ultima = None

        else:
            # Contar letras de la palabra
            letras_pal += 1
            # Identificar m y b a partir de la tercer letra
            if letras_pal >= 3:
                if letra == 'b':
                    tiene_b = True
                elif letra == 'm':
                    tiene_m = True

            # Detectar si comienza con p seguida de vocal
            if letras_pal == 1 and letra == 'p':
                tiene_p = True

            if letras_pal == 2 and es_vocal(letra) and tiene_p:
                tiene_pvocal = True

            # Guardar primera letra
            if letras_pal == 1:
                primera = letra
            ultima = letra

    # Resultados
```

```
    print('*' * 80)
    print('Palabras tienen una m y una b a partir de la tercer letra:', palabras_bym)
    print('Palabras que comienzan con la letra p seguida de cualquier vocal:', palabras_pvocal)
    print('Palabras que comienzan y terminan con el mismo carácter:', palabras_iniciofin)


# Script principal
test()
```

6. Las 2 vocales

Desarrollar un programa en Python que permita cargar por teclado un texto completo. Siempre se supone que el usuario cargará un punto para indicar el final del texto, y que cada palabra de ese texto está separada de las demás por un espacio en blanco. El programa debe

a) Determinar la cantidad de palabras en la que aparecen 2 vocales distintas

b) Determinar la cantidad de palabras cuya cantidad de letras es múltiplo de la cantidad de letras de la palabra anterior

c) Determinar la cantidad de palabras cuya cantidad de letras es superior a un valor ingresado por el usuario (debe ser mayor a cero)

d) Determinar la cantidad de palabras en las que su longitud en letras superan el promedio de letras del texto

6.1. Solución

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'


def es_vocal(caracter):
    vocales = 'aeiouáéíóú'
    return caracter in vocales


def leer_numero():
    numero = 0
    while numero <= 0:
        numero = int(input('Ingrese un numero: '))
        if numero <= 0:
            print('Error!!! el numero debe ser mayor a cero')
    return numero


def palabras_superan_promedio(texto, prom):
    cant_palabras = cant_letras = 0
    for caracter in texto:
        if caracter != ' ' and caracter != '.':
            cant_letras += 1
        else:
            if cant_letras > prom:
                cant_palabras += 1
    return cant_palabras


def es_multiplo(numero, divisor):
    resto = numero % divisor
    return resto == 0


def test():
    print('Analizador de Texto las 2 vocales')
    print('*' * 60)

    numero = leer_numero()
    texto = input('Ingrese el texto a analizar, finaliza con punto: ')

    pal_2voc_dif = cant_palabras = cant_letras = cant_let_pal_ant = pal_mult_ant = pal_let_mayor_num = total_letras = 0
    pal_sup_promedio = 0
    es_pri_vocal = True
    hay_2_voc_dif = False
    pri_vocal = ''

    for letra in texto:
        if letra == ' ' or letra == '.':
            if cant_letras > 0:

                cant_palabras += 1

                if hay_2_voc_dif:
                    pal_2voc_dif += 1

                if cant_palabras == 1:
                    cant_let_pal_ant = cant_letras
                else:
                    if es_multiplo(cant_letras, cant_let_pal_ant):
                        pal_mult_ant += 1
                    cant_let_pal_ant = cant_letras

                if cant_letras > numero:
                    pal_let_mayor_num += 1

                total_letras += cant_letras
```

```
            es_pri_vocal = True
            hay_2_voc_dif = False
            pri_vocal = ''
            cant_letras = 0

        else:
            cant_letras += 1

            if es_vocal(letra):
                if es_pri_vocal:
                    pri_vocal = letra
                    es_pri_vocal = False
                else:
                    if es_vocal(letra) and letra != pri_vocal:
                        hay_2_voc_dif = True
                        pri_vocal = ''

    if cant_palabras > 0:
        prom = total_letras // cant_palabras
        pal_sup_promedio = palabras_superan_promedio(texto, prom)

    print('La cantidad de palabras con 2 vocales diferentes son:', pal_2voc_dif)
    print('La cantidad de palabras cuya cantidad de letras es multiplo de la anterior son:', pal_mult_ant)
    print('Palabras que tienen mas de ', numero, 'letras son', pal_let_mayor_num)
    print('La cantidad de palabras que superan el promedio de letras de la palabra son', pal_sup_promedio)


test()
```

7. Dificultad = pow (parcial, 3)

Desarrollar un programa en Python que permita cargar por teclado un texto completo. Se supone que el usuario cargará un punto para indicar el final del texto, y que cada palabra de ese texto está separada de las demás por un espacio en blanco. El programa debe:

a) Determinar la cantidad de palabras que tuvieron exactamente 3 vocales.

b) Determinar el porcentaje de palabras que tuvieron algún dígito ('0' al '9') y más de 4 letras.

c) De las palabras que terminan con la primera letra de todo el texto, determinar el **orden** de la que tiene menor cantidad de caracteres. Por ejemplo, en el texto: 'Ana está en la casa', hay 4 palabras que terminan en la primera letra del texto ('Ana', 'está', 'la', 'casa'), la que menos caracteres tiene es: 'la' y su orden es 4 (porque es la cuarta palabra del texto).

d) Determinar la cantidad de palabras que contienen 'men' en la primera mitad de la palabra.

7.1. Solución anotada

```
__author__ = 'Cátedra de Algorítmos y Estructuras de datos'


def calcular_porcentaje(muestras, total):
    """
    Calcula el porcentaje que representan las muestras en el total
    :param muestras: La cuenta de muestras
    :param total: El total sobre el que se contaron las muestras
    :return: El porcentaje que representan las muestras en el total si total es distinto a 0.
    0 en caso contrario
    """
    porcentaje = 0
    if total != 0:
        porcentaje = (muestras * 100) / total
    return porcentaje


def es_vocal(caracter):
    """
    Comprueba si un caracter es una vocal o no
    :param caracter: El caracter que se quiere comprobar
    :return: True si es una vocal, False si no lo es
    """
    vocales = 'aeiouáéíóú'  # Vocales (en minúsculas)
    for vocal in vocales:
        if caracter == vocal:
            return True

    return False


def es_digito(caracter):
    """
    Comprueba si un caracter es un dígito o no
    :param caracter: El caracter a comprobar
    :return: True si es un dígito, False en caso contrario
    """
    if caracter >= '0' and caracter <= '9':
        return True
    return False


def main():
    """
    Función principal del programa. Toda la lógica del mismo está contemplada aquí
    :return: None
    """
    ########################################################################################################################
    # Variables Generales (Variables que no se van a reiniciar, son los resultados del programa)                           #
    ########################################################################################################################
    cont_palabras = 0  # Contador de palabras del texto
    # Punto 1
    cont_punto_1 = 0  # Contador de Palabras con 3 vocales
    # Punto 2
    cont_punto_2 = 0  # Contador de palabras con dígitos y más de 4 caracteres
    # Punto 3
    cant_caracteres_menor = None  # Cantidad de carac. de la menor palabra que termina con la primera letra del texto
    pos_menor_palabra = None  # Posición de la palabra con menos carc. entre las que termina con la primera letra
    # Punto 4
    cont_punto_4 = 0  # Contador de palabras que contienen 'men' en la primera mitad de la palabra

    ########################################################################################################################
    # Variables por palabra (Variables que tienen sentido palabra por palabra, se reinician ante un serparador)        #
    ########################################################################################################################
    cont_letras_palabra = 0  # Contador de letras de la palabra que se está analizando
    # Punto 1
    cont_vocales_palabra = 0  # Contador de vocales de la palabra que se está analizando
```

```
# Punto 2
b_tiene_digito = False  # Bandera que indica si una palabra tiene o no un dígito entre sus caracteres
# Punto 4
b_m = b_me = False      # Banderas que indican si se vió una 'm' o si se vió 'me'
b_tiene_men = False     # Bandera que indica si se encontró 'men' en la palabra
pos_silaba_men = 0      # Posición en la que termina la sílaba 'men' si se la encontró en la palabra

# Carga del texto
texto = input('Ingrese texto: (Terminar con "."): ')
# Conversión a minúsculas (Notar que esto no sirve si el enunciado solicita discriminar mayúsculas y minúsculas)
texto = texto.lower()

# Recorrido de la cadena
for car in texto:

    if car != ' ' and car != '.':
        # De este lado, estamos dentro de una palabra
        cont_letras_palabra += 1

        # Para el punto 1 -> Contar vocales
        if es_vocal(car):
            cont_vocales_palabra += 1
        else:
            # Para el punto 2 -> ver si es un dígito. Notar que esto está en el else.
            # Esto es así, porque si es una vocal, no va a ser un dígito...
            if es_digito(car):
                b_tiene_digito = True

        # Para el punto 3 -> determinar la primera letra del el texto. Para que se la primera
        # letra del texto, debe ser la primera palabra y el primer caracter de esa palabra.
        if cont_palabras == 0 and cont_letras_palabra == 1:
            primera_letra = car

        # Para el punto 4 -> Determinar si se encuentra 'men' y en qué posición de la palabra
        if car == 'm' and not b_tiene_men:
            # Si se encuentra una 'm' y todavía no se encontró 'men' en la palabra...
            b_m = True
        else:
            # Si se trata de una 'e' y en el caracter anterior se vió una 'm'
            if car == 'e' and b_m:
                # Si entramos aquí, ya vimos 'me'
                b_me = True
            else:
                # Si vemos 'n' y teníamos 'me'
                if car == 'n' and b_me:
                    # Entonces, vimos 'men'
                    b_tiene_men = True
                    # Pero también nos interesa la posición donde determinamos que vimos 'men'
                    pos_silaba_men = cont_letras_palabra
                # Se baja la bandera b_me, para evitar que quede levantada si hemos visto 'me'
                # pero no 'men'
                b_me = False
            # Se baja la bandera b_m. Esto es para evitar que quede levantada la bandera cuando
            # no se vió 'me'
            b_m = False

        # Para el punto 3 -> se almacena el último caracter, para saber con qué termina la palabra.
        # Se hace al final del if por una cuestión de orden, en este caso pudo estar antes
        ultimo_car = car

    else:
        # En esta rama, estamos ante la posible terminación de una palabra
        if cont_letras_palabra > 0:
            # Únicamente se cuenta una palabra (y se la procesa) si consta de más de un caracter
            cont_palabras += 1

            # Para el punto 1 -> Si tiene exactamente 3 vocales, se la cuenta
```

```
                if cont_vocales_palabra == 3:
                    cont_punto_1 += 1

                # Para el punto 2 -> Si tiene un dígito y más de 4 letras
                if b_tiene_digito and cont_letras_palabra > 4:
                    cont_punto_2 += 1

                # Para el punto 3 -> Si termina con la primera letra del texto
                if ultimo_car == primera_letra:
                    # Pero como piden saber, dentro de todas las palabras que terminan con la primera letra
                    # del texto, cuál es la posición de la de menor cantidad de caracteres...
                    if cant_caracteres_menor is None or cont_letras_palabra < cant_caracteres_menor:
                        # Como en cualquier búsqueda de menor, nos quedamos con la cantidad de caracteres
                        # y la posición de la palabra (que es lo que nos piden)
                        cant_caracteres_menor = cont_letras_palabra
                        pos_menor_palabra = cont_palabras

                # Para el punto 4 -> Si contiene 'men' en la primera mitad de la palabra
                if b_tiene_men and pos_silaba_men <= (cont_letras_palabra / 2):
                    cont_punto_4 += 1

            # Reseteo de las variables que tienen que ver con cada palabra
            cont_letras_palabra = 0
            cont_vocales_palabra = 0
            b_tiene_digito = False
            pos_silaba_men = 0
            b_tiene_men = False

    # Terminó el for, aquí se procesan los resultados
    porc_punto_2 = calcular_porcentaje(cont_punto_2, cont_palabras)
    print('============================== RESULTADOS ==============================')
    print('Total de palabras procesadas: ', cont_palabras)
    print('1) Cantidad de palabras con 3 vocales: ', cont_punto_1)
    print('2) Procentaje de palabras con algún dígito y más de 4 letras: ', round(porc_punto_2, 2))
    if pos_menor_palabra is not None:
        print('3) Posición de la menor palabra que termina con la primera letra del texto: ', pos_menor_palabra)
    else:
        print('3) No hubo palabras que terminasen con la primera letra del texto')
    print('4) La cantidad de palabras que contienen "men" en la primera mitad de la palabra es: ', cont_punto_4)


if __name__ == '__main__':
    main()
```

8. Entidad Bancaria

Una entidad bancaria necesita hacer una evaluación sobre las operatorias que han realizado sus clientes, para ello pide un programa que permita determinar dichas estadísticas.

Para ejecutarse correctamente el programa debe validar un token que cumple con las siguientes características:

- Debe contener dos letras mayúsculas
- Debe contener al menos 2 dígitos
- Debe incluir el signo numeral en la segunda mitad del token

Si cumple el programa deberá pedir la cantidad de operaciones bancarias a procesar, ingresando en forma aleatoria para cada una de ellas:

- Si fue una empresa o un particular
- El monto que se opero
- Si opera en pesos o en dólares

Luego presentar un menú de opciones que permita informar:

1 - Cantidad operaciones que se llevaron a cabo

2 - Porcentaje de operaciones en dólares

3 - Monto total operado en pesos por particulares

4 - Monto promedio operado en pesos por las empresas

8.1. Solución

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'

import random


def es_mayuscula(letra):
    return letra >= 'A' and letra <= 'Z'


def es_digito(car):
    digitos = '0123456789'
    return car in digitos


def validar_token(token):
    cant_may = 0
    cant_dig = 0
    pos = 0
    pos_numeral = 0
    for car in token:
        if es_mayuscula(car):
            cant_may += 1
        elif es_digito(car):
            cant_dig += 1
        elif car == '#':
            pos_numeral = pos
        pos += 1
    if cant_may == 2 and cant_dig >= 2 and pos_numeral > len(token) // 2:
        return True
    else:
        return False


def validar_positivo(mensaje):
    num = int(input(mensaje))
    while num <= 0:
        print('Inválido! Debe ser un valor mayor que cero')
        num = int(input(mensaje))
    return num


def generar_datos():
    tipos = 'Particular', 'Empresa'
    monedas = 'Pesos', 'Dolares'
    tipo = random.choice(tipos)
    monto = round(random.uniform(100, 100000), 2)
    moneda = random.choice(monedas)
    return tipo, monto, moneda


def calcular_porcentaje(cant, total):
    if total != 0:
        return cant * 100 / total
    else:
        return 0


def calcular_promedio(suma, cant):
    if cant != 0:
        return suma / cant
    else:
        return 0


def mostrar_menu(n, monto_particulares, porc_dolares, prom_empresas):
    opcion = -1
    while opcion != 0:
```

```
        print('-' * 60)
        print('Menu de Opciones')
        print('1 - Cantidad de operaciones')
        print('2 - Porcentaje de operaciones en dolares')
        print('3 - Monto total operado en pesos por particulares')
        print('4 - Monto promedio operado por empresas')
        print('0 - Salir')
        opcion = int(input('Ingrese su opcion: '))
        if opcion == 1:
            print('Total de operaciones =', n)
        elif opcion == 2:
            print('Porcentaje de operaciones en dolares =', round(porc_dolares, 2))
        elif opcion == 3:
            print('Monto total operado en pesos por particulares =', round(monto_particulares, 2))
        elif opcion == 4:
            print('Monto promedio operado en pesos por empresas =', round(prom_empresas, 2))
        print('-' * 60)


def principal():
    print('=' * 20, 'OPERACIONES BANCARIAS', '=' * 20)
    # SUB1 : Validar token
    token = input('Ingrese su token: ')
    if validar_token(token):
        # SUB2: Procesar operaciones
        n = validar_positivo('Ingrese cantidad de operaciones: ')
        ops_dolares = 0
        monto_particulares = 0
        monto_empresas = 0
        ops_empresas = 0
        for i in range(n):
            # Generar datos
            tipo, monto, moneda = generar_datos()
            print('Cliente', tipo, ' - Monto', monto, 'en', moneda)
            # Procesar
            if moneda == 'Dolares':
                ops_dolares += 1
            if moneda == 'Pesos':
                if tipo == 'Particular':
                    monto_particulares += monto
                elif tipo == 'Empresa':
                    monto_empresas += monto
                    ops_empresas += 1
        # Resultados
        porc_dolares = calcular_porcentaje(ops_dolares, n)
        prom_empresas = calcular_promedio(monto_empresas, ops_empresas)
        # SUB3: MOSTRAR MENU
        mostrar_menu(n, monto_particulares, porc_dolares, prom_empresas)
    else:
        print('El token ya no es valido')


principal()
```