# Guía de Ejercicios Prácticos - Ficha 09

Sitio: Universidad Virtual UTN FRC
Curso: Algoritmos y Estructuras de Datos (2020)
Libro: Guía de Ejercicios Prácticos - Ficha 09
Imprimido por: Luciana Lisette Montarce
Día: lunes, 23 de noviembre de 2020, 21:19

# Descripción

Esta guía contiene enunciados de algunos ejercicios para aplicar los conceptos de programación en ***Python*** que se analizan en la ***Ficha 09***. 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. Secuencia 1, 2, 3

Desarrollar un programa en Python que permita cargar por teclado una sucesión 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 cuántos de los números ingresados eran divisibles por 4.

b) Determinar el mayor de los números impares ingresados.

c) Determinar cuántas veces se ingresó el primero de los números (cuente al primero).

d) Determinar cuántas veces se ingresó un 1, seguido de un 2, y seguido a su vez de un 3. Por ejemplo: en la sucesión: 3, 6, 1, 2, 3, 7, 8, 2, 3, 1, 2, 3, 0 ocurrió dos veces.

## 1.1. Solución

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

# titulo general...
print('Deteccion de secuencia 1-2-3')
print('Carga de la secuencia, terminando en 0(cero)...')

# inicializacion de flags y contadores...
# contador de números ingresados...
cn = 0

# contador de números divisibles por 4...
cd4 = 0

# contador de veces que entro 1-2-3...
c123 = 0

# el mayor de los impares...
myi = None

# el primero de todos los numeros...
prim = None

# cuantas veces entro el primero de todos?...
cvp = 0

# flag: paso el primero de los impares?...
spi = False

# flag: el ultimo fue un 1?...
s1 = False

# flag: el ultimo fue un 2 e inmediatamente paso un 1?...
s12 = False

# carga del primer numero...
num = int(input('Numero (ingrese un 0(cero) para terminar): '))

# ciclo de carga para procesar le secuencia...
while num != 0:
    # contar el número actual...
    cn += 1

    # 1.) contar divisibles por 4...
    if num % 4 == 0:
        cd4 += 1

    # 2.) determinar el mayor de los impares...
    if num % 2 == 1:
        if not spi:
            myi = num
            spi = True
        elif num > myi:
            myi = num

    # 3.) determinar cuantas veces entro el primero de todos...
    if cn == 1:
        prim = num
    if num == prim:
        cvp += 1

    # 4.) determinar cuantas veces entro la secuencia 1-2-3...
    if num == 1:
        s1 = True
```

```
        s12 = False
    else:
        if num == 2 and s1:
            s12 = True
        else:
            if num == 3 and s12:
                c123 += 1

            s12 = False
        s1 = False

    # carga del siguiente numero...
    num = int(input('Otro (ingrese un 0(cero) para terminar): '))

# analisis de los resultados finales...
print('1. Cantidad de numeros divisibles por 4:', cd4)
print('2. Mayor de los impares:', myi)
print('3. El primero fue el', prim, 'y se ingreso:', cvp, 'veces')
print('4. Cantidad de veces que se formo la secuencia 1-2-3:', c123)
```

## 1.2. Solución con funciones

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


def test():
    # titulo general...
    print('Deteccion de secuencia 1-2-3')
    print('Carga de la secuencia, terminando en 0(cero)...')

    # inicializacion de flags y contadores...
    # contador de números ingresados...
    cn = 0

    # contador de números divisibles por 4...
    cd4 = 0

    # contador de veces que entro 1-2-3...
    c123 = 0

    # el mayor de los impares...
    myi = None

    # el primero de todos los numeros...
    prim = None

    # cuantas veces entro el primero de todos?...
    cvp = 0

    # flag: paso el primero de los impares?...
    spi = False

    # flag: el ultimo fue un 1?...
    s1 = False

    # flag: el ultimo fue un 2 e inmediatamente paso un 1?...
    s12 = False

    # carga del primer numero...
    num = int(input('Numero (ingrese un 0(cero) para terminar): '))

    # ciclo de carga para procesar le secuencia...
    while num != 0:
        # contar el número actual...
        cn += 1

        # 1.) contar divisibles por 4...
        if num % 4 == 0:
            cd4 += 1

        # 2.) determinar el mayor de los impares...
        if num % 2 == 1:
            if not spi:
                myi = num
                spi = True
            elif num > myi:
                myi = num

        # 3.) determinar cuantas veces entro el primero de todos...
        if cn == 1:
            prim = num
        if num == prim:
            cvp += 1

        # 4.) determinar cuantas veces entro la secuencia 1-2-3...
```

```
        if num == 1:
            s1 = True
            s12 = False
        else:
            if num == 2 and s1:
                s12 = True
            else:
                if num == 3 and s12:
                    c123 += 1

                s12 = False
            s1 = False

        # carga del siguiente numero...
        num = int(input('Otro (ingrese un 0(cero) para terminar): '))

    # analisis de los resultados finales...
    print('1. Cantidad de numeros divisibles por 4:', cd4)
    print('2. Mayor de los impares:', myi)
    print('3. El primero fue el', prim, 'y se ingreso:', cvp, 'veces')
    print('4. Cantidad de veces que se formo la secuencia 1-2-3:', c123)


# script principal...
# ... solo invocar a test()...
test()
```

# 2. Secuencia 5, 5

Desarrollar un programa en Python que permita cargar por teclado una sucesión 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 cuántos de los números ingresados eran mayores que *n1* y menores que *n2* (cargar previamente por teclado los números *n1* y *n2*).

b) Determinar el porcentaje que el total de números calculado en el punto 1 representa en el total de números cargados.

c) Determinar si en algún momento se ingresó un número impar seguido inmediatamente de un par. No importa cuántas veces ocurrió: sólo indicar si al menos una vez pasó.

d) Determinar cuántas veces se ingresó un 5 seguido inmediatamente de otro 5. Por ejemplo: en la sucesión: 3, 6, 5, 5, 7, 5, 2, 5, 9, 5, 5, 0 ocurrió dos veces.

## 2.1. Solución

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

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


def es_par(numero):
    resto = numero % 2
    if resto == 0:
        return True
    else:
        return False

def test():
    print('Analisis de Secuencia 5,5')
    print('-' * 80)

    n1 = int(input('Ingrese la cota minima del intervalo a definir: '))
    n2 = int(input('Ingrese la cota maxima del intervalo a definir: '))

    cant_num_intervalo = cant_numeros = cant_secuencia_5 = 0
    hay_par_impar = False
    anterior = 0
    numero = int(input('Ingrese un numero de la secuencia, con cero termina: '))
    while numero != 0:
        cant_numeros += 1
        if n1 < numero < n2:
            cant_num_intervalo += 1

        if cant_numeros > 1:
            if not es_par(anterior) and es_par(numero):
                hay_par_impar = True

            if anterior == 5 and numero == 5:
                cant_secuencia_5 += 1
        anterior = numero
        numero = int(input('Ingrese otro numero de la secuencia: '))

    porc = calcular_porcentaje(cant_num_intervalo, cant_numeros)
    print('La cantidad de numeros dentro del intevalo [', n1, ':', n2, '] es: ', cant_num_intervalo, sep='')
    print('El porcentaje de numero en el intervalo sobre el total de numeros es: ', porc, '%', sep='')
    print('La cantidad de secuencia 5,5 ingresada es:', cant_secuencia_5)
    if hay_par_impar:
        print('Se ha ingreseado al menos un numero impar seguido de un par')

test()
```

# 3. Inicio con sílaba "pa"

Desarrollar un programa en Python que permita cargar por teclado un texto. Siempre se supone que 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 determinar:

a) La cantidad de palabras que comienzan con la expresión "pa" y termina con la letra "n"

c) La cantidad de palabras que presentan mas de dos vocales a partir de la tercera letra de la palabra

d) El porcentaje que representa la cantidad de palabras del punto anterior respecto de la cantidad de total de palabras del texto

e) Cantidad de palabras de mas de 5 letras

## 3.1. Solucion

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

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

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

def main():
    print('Analisis de Texto - Silabas \"pa\"')
    print('-' * 80)

    cont_letras = cant_pal_pa = cant_vocales = cant_pal_dosvocales = 0
    cant_palabras = cant_pal_5letras = 0

    tiene_p = tiene_pa = tiene_n = False
    texto = input('Ingrese el texto a analizar, debe finalizar con punto: ')
    for letra in texto:
        if letra == ' ' or letra == '.':
            if cont_letras > 0:
                cant_palabras += 1
                if tiene_pa and tiene_n:
                    cant_pal_pa += 1

                if cant_vocales > 2:
                    cant_pal_dosvocales += 1

                if cont_letras > 5:
                    cant_pal_5letras += 1

            tiene_p = False
            tiene_pa = False
            cant_vocales = 0
            cont_letras = 0
        else:
            cont_letras += 1
            if cont_letras == 1 and letra == 'p':
                tiene_p = True
            else:
                if tiene_p and letra == 'a':
                    tiene_pa = True
                tiene_p = False

            if cont_letras > 3:
                if es_vocal(letra):
                    cant_vocales += 1

            tiene_n = False
            if letra == 'n':
                tiene_n = True

    porcentaje = calcular_porcentaje(cant_pal_pa, cant_palabras)
    print('La cantidad de palabras que empiezan con \"pa\" y '
          'terminan con \"s\" es:', cant_pal_pa)
    print('El porcentaje de la cantidad anterior sobre el total '
          'del texto es: ', porcentaje, '%', sep='')
    print('La cantidad de palabras con mas de 2 vocales a partir '
          'de la tercera letra es:', cant_pal_dosvocales)
    print('La cantidad de palabras con mas de cinco letras es:',
          cant_pal_5letras)

main()
```

# 4. Secuencia en rango

Desarrollar un programa de Python 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 cuantos números se encuentran en el rango definido por 2 números p y q previamente ingresados (validar que los números que definen el rango son mayores a cero)

b) Determinar la cantidad de veces que se ingresaron 2 números contiguos pares.

c) Determinar la cantidad de números que son múltiplos del primer numero de la secuencia

d) Determinar el porcentaje que representa la cantidad del primer punto sobre el total de números de la secuencia

## 4.1. Solución

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


def validar_mayor(valor, mensaje='Ingrese un numero: '):
    numero = 0
    while numero <= valor:
        numero = int(input(mensaje))
        if numero <= valor:
            print('Error!!!! El numero debe ser mayor a ', valor)
    return numero


def porcentaje(total, cantidad):
    por = 0
    if total != 0:
        por = round(cantidad * 100 / total, 2)
    return por


def validar_rango():
    cota_inferior = validar_mayor(0, 'Ingrese la cota inferior de rango: ')
    cota_superior = validar_mayor(0, 'Ingrese la cota superior de rango: ')
    while cota_superior < cota_inferior:
        cota_superior = validar_mayor('Ingrese la cota superior de rango, debe ser mayor a la inferior: ')
    return cota_inferior, cota_superior


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


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


def test():

    print('Secuencia de Rangos')
    print('*' * 60)

    primer_numero = 0
    anterior = -1
    primero = True
    cantidad_rango = cant_pares_contig = cant_mult = cant_num = 0

    cota_inferior, cota_superior = validar_rango()

    numero = int(input('Ingresar un numero, la secuencia finaliza cuando ingrese el: '))
    while numero != 0:

        cant_num += 1
        if primero:
            primer_numero = numero
            primero = False
        else:
            if es_multiplo(numero, primer_numero) == 0:
                cant_mult += 1

        if cota_inferior <= numero <= cota_superior:
            cantidad_rango += 1

        if es_par(anterior) and es_par(numero) == 0:
            cant_pares_contig += 1

        anterior = numero
        numero = int(input('Ingrese el siguiente numero de la secuencia: '))
```

```
    print('-' * 60)
    print('Resultados')
    print('-' * 60)

    porc = porcentaje(cant_num, cantidad_rango)
    print('Hay', cantidad_rango, 'números se encuentran en el rango definido entre', cota_inferior, 'y', cota_superior)
    print('Y representa un', porc, '% del total de numeros de la serie')
    print(cant_pares_contig, 'veces que se ingresaron 2 números contiguos pares')
    print(cant_mult, 'números que son múltiplos del primer numero de la secuencia ')


test()
```

# 5. Sílaba "pe"

Se pide desarrollar un programa en Python que permita cargar por teclado un texto completo en una variable de tipo cadena de caracteres. 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 tienen 3, 5 o 7 letras.

b) Determinar la cantidad de palabras con más de tres letras, que tienen una vocal en la tercera letra.

c) Determinar el porcentaje de palabras que tienen sólo una o dos vocales sobre el total de palabras del texto.

d) Determinar la cantidad de palabras que contienen más de una vez la sílaba "pe".

## 5.1. Solución

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

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


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


def main():
    print('Analizar texto con letras pe')
    print('-' * 80)

    texto = input('Ingrese el texto a analizar, finaliza con punto: ')
    cont_letras = cont_palabras = cont_pal3letras = pal_vocales_tercer_letra = cant_vocales = pal_2vocales = 0
    cant_pe = palabras_mas2_pe = 0
    anterior = ''
    tiene_vocal = False

    for letra in texto:
        if letra == ' ' or letra == '.':
            if cont_letras > 0:
                cont_palabras += 1
                if cont_letras == 3 or cont_letras == 5 or cont_letras == 7:
                    cont_pal3letras += 1

                if cont_letras > 3 and tiene_vocal:
                    pal_vocales_tercer_letra += 1

                if cant_vocales <= 2:
                    pal_2vocales += 1

                if cant_pe >= 2:
                    palabras_mas2_pe += 1

            cant_pe = 0
            cant_vocales = 0
            tiene_vocal = False
            cont_letras = 0
        else:
            cont_letras += 1
            if es_vocal(letra):
                cant_vocales += 1
                if cont_letras == 3:
                    tiene_vocal = True

            if letra == 'e' and anterior == 'p':
                cant_pe += 1

        anterior = letra

    porc = calcular_porcentaje(pal_2vocales, cont_palabras)
    print('La cantidad de palabras con 3, 5 o 7 letras de longitud son:', cont_pal3letras)
    print('La cantidad de palabras con mas de 3 letras y una vocal en la tercer letra son:', pal_vocales_tercer_letra)
    print('El porcentaje de palabras con 2 vocales sobre el total del texto es: ', porc, '%', sep='')
    print('La cantidad de palabras con mas de una sílaba \'pe\' es:', palabras_mas2_pe)

main()
```

# 6. Secuencia promedio de pares

Se pide desarrollar un programa en Python controlado por un menú de opciones. Ese menú debe permitir gestionar las siguientes tareas, siempre usando funciones que acepten parámetros y/o retornen valores en cada situación en que se considere apropiado:

a) Cargar una sucesión de numeros enteros y positivos (validar que efectivamente cada número que se cargue sea mayor a cero). La carga finaliza cuando se ingresa un cero. Mostrar la cantidad de numeros de la sucesión cargada que son multiplos del primer numero de la secuencia.

b) Ingresar por teclado dos números $p$ y $q$ que definen los límites de un intervalo $[p, q]$ (validar que $0 < p < q$) y una sucesión de $n$ números (también cargando $n$ por teclado y validando que $n > 0$) Determine cuántos de los números de la sucesión cargada están fuera del intervalo y cuántos están dentro del intervalo, e indicar también cuántos de los números cargados eran pares y estaban dentro del intervalo dado.

c) Cargar una secuencia de números enteros positivos (validar que efectivamente cada número que se cargue sea mayor a cero). La carga debe terminar cuando se ingrese un número mayor a 100. Determine si en la secuencia se ingresaron dos números contiguos que sean pares. Si es así muestre el promedio de todos los números pares que se hayan ingresado. Si no, informe que no se ingresaron números contiguos pares. Ejemplo: en la secuencia {1, 2, 3, 6, 4, 7, 8} hay al menos un dos números contiguos que son pares (el 6 y el 4) y en este caso, el promedio de todos los pares de la secuencia es (2 + 6 + 4 + 8) / 4 = 20/4 = 5. Pero en esta secuencia: {1, 2, 5, 7, 8, 9, 3, 6, 1} no hay números contiguos pares.

## 6.1. Solucion

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


def validar_mayor(valor, mensaje):
    numero = int(input(mensaje))
    while numero <= valor:
        print('Error!!! El numero debe ser mayor a', valor)
        numero = int(input(mensaje))

    return numero


def es_multiplo(numero, divisor):
    paridad = numero % divisor
    if paridad == 0:
        return True
    else:
        return False


def multiplos_primer_valor():
    cantidad = 0
    numero = validar_mayor(-1, 'Ingrese un numero de la secuencia (finaliza con cero): ')
    es_primero = True
    primero = -1
    while numero != 0:
        if es_primero:
            es_primero = False
            primero = numero
        else:
            if es_multiplo(numero, primero):
                cantidad += 1
        numero = validar_mayor(-1, 'Ingrese otro numero de la secuencia: ')
    return cantidad

def leer_intervalo():
    p = validar_mayor(0, 'Ingrese la cota inferior del intervalo: ')
    q = validar_mayor(0, 'Ingrese la cota superior del intervalo: ')
    while q < p:
        print('El Intervalo definido no es correcto, reingrese los valores')
        p = validar_mayor(0, 'Ingrese la cota inferior del intervalo: ')
        q = validar_mayor(0, 'Ingrese la cota superior del intervalo: ')
    return p, q


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


def analizar_intervalo(p, q, n):
    can_fuera = can_dentro, can_pares = 0
    for i in range(n):
        numero = int(input('Ingrese un numero a analizar: '))
        if numero < p or numero > q:
            can_fuera += 1
        else:
            can_dentro += 1
            if es_par(numero):
                can_pares += 1

    return can_fuera, can_dentro, can_pares


def calcular_promerio(dividendo, divisor):
    prom = 0
    if divisor != 0:
        prom = dividendo / divisor
```

```
    return round(prom, 2)


def analizar_pares_contiguos():
    numero = validar_mayor(0, 'Ingrese un numero de la secuencia: ')
    anterior = -1
    hay_pares_contiguos = False
    acumulador = contador = 0
    while numero < 100:
        if es_par(numero):
            contador += 1
            acumulador += numero

        if es_par(anterior) and es_par(numero):
            hay_pares_contiguos = True

        anterior = numero
        numero = validar_mayor(0, 'Ingrese un numero de la secuencia: ')

    prom = 0
    if hay_pares_contiguos:
        prom = calcular_promerio(acumulador, contador)
    return hay_pares_contiguos, prom

def test():
    menu = 'Menu de Opciones\n' \
           '1 - Secuencia de multiplos\n' \
           '2 - Analisis de rango\n' \
           '3 - Promedio de Pares\n' \
           '4 - Salir\n'\
           'Ingrese su opcion: '

    opcion = - 1
    while opcion != 4:
        opcion = int(input(menu))
        if opcion == 1:
            cant_numero = multiplos_primer_valor()
            print('La cantidad de numeros multiplos del primero son:', cant_numero)
        elif opcion == 2:
            p, q = leer_intervalo()
            n = validar_mayor(0, 'Ingrese la cantidad de numeros a procesar: ')

            cant_fuera, cant_dentro, cant_pares_dentro = analizar_intervalo(p, q, n)

            print('La cantidad de valores fuera de [', p, ':', q, '] son: ', cant_fuera, sep='')
            print('La cantidad de valores dentro de [', p, ':', q, '] son: ', cant_dentro, sep='')
            print('La cantidad de valores pares dentro de [', p, ':', q, '] son: ',
                  cant_pares_dentro, sep='')
        elif opcion == 3:
            hay_pares, promedio = analizar_pares_contiguos()
            if hay_pares:
                print('En la secuencia hubo pares contiguos y su promedio es:', promedio)
            else:
               print('En la secuencia no hubo pares contiguos')


test()
```

# 7. Menú de repetitivas

Desarrollar un programa controlado por menú de opciones, que permita:

a) Tablas: ingresar por teclado un valor positivo entre 1 y 10 (validarlo). Mostrar la tabla de multiplicar correspondiente a dicho número. (Por ej: si se ingresa 3, mostrar 3x1=3, 3x2=6 etc.)

b) Mayor y menor: ingresar por teclado una sucesión de números positivos (validar), que termina cuando se ingresa 0. Informar mayor y menor valor de la sucesión.

c) Múltiplos: ingresar por teclado dos valores a y b, siendo a positivo y b mayor que a (validarlo). Sumar todos los múltiplos de a comprendidos en el intervalo [a,b]

d) Texto: ingresar por teclado un texto terminado en un punto (validar esto), cuyas palabras se separan por un único espacio. Informar cuántas palabras terminan en vocal y qué porcentaje representan sobre el total de palabras del texto.

## 7.1. Solución

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


def validar_entre(minimo, maximo):
    mensaje = 'Ingrese un valor entre ' + str(minimo) + ' y ' + str(maximo) + ': '
    valor = int(input(mensaje))
    while valor <= minimo or valor >= maximo:
        valor = int(input('Inválido! ' + mensaje))
    return valor


def validar_mayor_que(minimo):
    mensaje = 'Ingrese un valor mayor que ' + str(minimo) + ': '
    valor = int(input(mensaje))
    while valor <= minimo:
        valor = int(input('Inválido! ' + mensaje))
    return valor


def validar_punto_final():
    mensaje = 'Ingrese un texto terminado en punto: '
    texto = input(mensaje)
    while texto[-1] != '.':
        texto = input('Inválido! ' + mensaje)
    return texto


def impares():
    num = validar_entre(0,11)
    for i in range(11):
        print(num,'x',i,'=',num*i)

def mayor_y_menor():
    num = validar_mayor_que(-1)
    primero = True
    may = 0
    men = 0
    while num!= 0:
        if primero:
            may = num
            men = num
            primero = False
        else:
            if num > may:
                may = num
            if num < men:
                men = num
        num = validar_mayor_que(-1)
    return may, men


def multiplos():
    a = validar_mayor_que(0)
    b = validar_mayor_que(a)
    suma = 0
    for multiplo in range(a,b,a):
        suma += multiplo
    return suma


def calcular_porcentaje(valor, total):
    if total == 0:
        return 0
    else:
        return round(valor * 100 / total, 2)

def texto():
```

```
    texto = validar_punto_final()
    #Procesar texto
    vocales = 'aeiou'
    pal_vocal = 0
    palabras = 0
    anterior = ''
    for letra in texto:
        if letra == '.' or letra == ' ':
            if anterior != '':
                palabras += 1
            if anterior in vocales:
                pal_vocal += 1
        else:
        anterior = letra
    #Calcular porcentaje
    porc = calcular_porcentaje(pal_vocal,palabras)
    return pal_vocal, porc


def  test():
    opcion = -1
    while opcion != 0:
        print('*'*80)
        print('OPCIONES REPETITIVAS')
        print('1-Impares')
        print('2-Mayor y menor')
        print('3-Múltiplos')
        print('4-Texto')
        print('0-Salir')
        opcion = int(input('\nIngrese su opción: '))
        if opcion == 1:
            impares()
        elif opcion == 2:
            may, men = mayor_y_menor()
            print('El mayor valor de la serie fue',may)
            print('El menor valor de la serie fue',men)
        elif opcion == 3:
            suma = multiplos()
            print('La suma de los múltiplos es',suma)
        elif opcion == 4:
            pal, porc = texto()
            print('Se detectaron',pal,'palabras con vocal, y son el',porc,'del texto')
        print('*'*80)


test()
```

# 8. Juego de Cartas

Desarrollar un programa para implementar un juego de cartas de la baraja española. Es una competencia de n rondas entre 2 jugadores (n se carga por teclado, validar)

En cada ronda, cada jugador recibe una carta (cuyo número y palo el programa deberá generar de forma aleatoria) y se define la ronda de la siguiente manera:

- El jugador que tenga la carta de mayor valor, se lleva ambas.
- Si las cartas son del mismo valor, se las lleva quien tenga una carta de oro.
- Si ambos tienen oro o ninguno lo tiene, cada jugador recupera su carta.

Los puntos de cada jugador se determinan sumando los valores de todas las cartas que ganó. Será triunfador el que tenga mayor puntaje total.

## 8.1. Solucion

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

import random

def validar_mayor_que(minimo, mensaje):
    num = int(input(mensaje))
    while num <= minimo:
        num = int(input('INVALIDO! ' + mensaje))
    return num


def generar_carta():
    palos = ('Copa', 'Basto', 'Espada', 'Oro')
    num = random.randint(1, 12)
    palo = random.choice(palos)
    return num,palo


def jugar(n):
    puntos1 = puntos2 = 0
    for ronda in range(n):
        print('\nRONDA',ronda)

        carta1 = generar_carta()
        print('Jugador 1 obtiene', carta1)

        carta2 = generar_carta()
        print('Jugador 2 obtiene', carta2)

        suma = carta1[0] + carta2[0]
        if carta1[0] > carta2[0]:
            puntos1 += suma
        elif carta2[0] > carta1[0]:
            puntos2 += suma
        else:
            if carta1[1] == 'Oro' and carta2[1]!='Oro':
                puntos1 += suma
            elif carta1[1] != 'Oro' and carta2[1]=='Oro':
                puntos2 += suma
            else:
                puntos1 += carta1[0]
                puntos2 += carta2[0]
        print('Puntajes: ',puntos1,'a',puntos2)

    return puntos1,puntos2


def mostrar_resultado(puntos1, puntos2):
    if puntos1 > puntos2:
        print('¡El Jugador 1 es el ganador!')
    elif puntos2 > puntos1:
        print('¡El Jugador 2 es el ganador!')
    else:
        print('¡Los jugadores empataron!')


def principal():
    print('JUEGO DE CARTAS')
    print('*' * 80)
    n = validar_mayor_que(0,'Ingrese rondas a jugar: ')
    puntos1, puntos2 = jugar(n)
    print('*' * 80)
    mostrar_resultado(puntos1,puntos2)


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

# 9. Portal de empleo

Un conocido portal de empleo requiere un programa para validar las búsquedas que se cargan en su página. Por cada búsqueda se requiere:

- CUIT: validar que sea un texto compuesto por 13 números separados por guiones de la siguiente manera: 00-00000000-0
- Descripción de la búsqueda: un texto donde cada palabra se separa con un espacio y termina con punto. Debe tener un máximo de 60 caracteres y un mínimo de 3 palabras. Ninguna palabra debe contener dos mayúsculas seguidas.
- Salario ofrecido: controlar que sea un valor mayor a 0

Si todos los datos son válidos, mostrar el aviso completo. En caso contrario, informar que no es posible mostrarlo.

Para terminar, consultar al usuario si desea cargar otro aviso o salir del programa.

## 9.1. Solucion

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

def validar_cuit(cuit):
    # CUIT: un texto compuesto por 13 números
    # separados por guiones de la siguiente manera: 00-00000000-0
    valido = False
    cant_numeros = 0
    if len(cuit) == 13:
        if cuit[2] == '-' and cuit[11] == '-':
            for car in cuit:
                if car in '0123456789':
                    cant_numeros += 1
            # Controlo si todos los que no son guiones, son digitos
            if cant_numeros == 11:
                valido = True
    return valido


def es_mayuscula(letra):
    if letra >= 'A' and letra <= 'Z':
        return True
    else:
        return False


def validar_descripcion(descripcion):
    # Descripción de la búsqueda: un texto donde
    # cada palabra se separa con un espacio y termina con punto.
    # Debe tener un máximo de 60 caracteres y un mínimo de 3 palabras.
    # Ninguna palabra debe contener 2 mayúsculas seguidas.
    valida = False
    cant_palabras = 0
    anterior = ''
    tiene_mayusc_seguidas = False
    if len(descripcion) <= 60:
        for letra in descripcion:
            # Contamos las palabras
            if letra == ' ' or letra == '.':
                cant_palabras += 1
            else:
                # Controlamos mayusculas seguidas
                if es_mayuscula(letra) and es_mayuscula(anterior):
                    tiene_mayusc_seguidas = True
            anterior = letra
        # Controlamos cantidad de palabras
        if cant_palabras >= 3:
            # Controlamos que no haya mayusculas seguidas
            if tiene_mayusc_seguidas == False:
                valida = True
    return valida


def validar_salario(salario):
    # Salario ofrecido: un valor mayor a 0
    if salario > 0:
        valido = True
    else:
        valido = False
    return valido

def principal():
    rta = 'S'
    while rta == 'S':
        print('PORTAL DE EMPLEOS')
        # Datos
        cuit_empleador = input('Ingrese cuit: ')
        # Proceso
```

```
        if validar_cuit(cuit_empleador):
            descripcion = input('Ingrese descripcion de la busqueda: ')
            if validar_descripcion(descripcion):
                salario = int(input('Ingrese salario ofrecido (mayor a 0): '))
                if validar_salario(salario):
                    # Mostramos el aviso
                    print('=' * 50)
                    print('AVISO')
                    print('El empleador', cuit_empleador, 'te esta buscando!')
                    print('Busqueda:', descripcion)
                    print('El salario ofrecido es de $', salario)
                    print('=' * 50)
                else:
                    print('El salario no es valido')
            else:
                print('La descripcion no es valida')
                print('Debe tener un máximo de 60 caracteres y un mínimo de 3 palabras.')
                print('Ninguna palabra debe contener 2 mayúsculas seguidas.')
        else:
            print('El cuit no es valido')
            print('Debe ser un texto compuesto por 13 números separados por guiones')
            print('de la siguiente manera: 00-00000000-0')
        rta = input('Quiere cargar otro aviso? (S/N) ')

    print('Programa terminado')


principal()
```