TEC – Escuela de Computación – Taller de programación – TAREA 3

PROGRAMACIÓN ESTRUCTURADA
PRÁCTICA USANDO ESTRUCTURAS DE SECUENCIA Y CONDICIONAL (if)

OBJETIVO DE LA PRÁCTICA: aplicación intensiva del if.

INSTRUCCIONES:

- Desarrollar en Python 3 en un archivo de nombre: tarea3_su_nombre.py.
- Use el material estudiado a la fecha. Solo se permite usar el tipo de datos numérico (int o float) y el booleano. Puede usar strings para interfaz con usuario. Tampoco se permite el uso de estructuras de repetición, no son necesarias y por otro lado el objetivo es practicar la estructura condicional.
- Los datos de entrada siempre serán numéricos.
- Probar la ejecución de cada función desde el modo comando con diferentes valores. Los programas deben ser generalistas, es decir, funcionar con cualquier grupo de datos que cumplan con las especificaciones.
- Enviar la solución al tecDigital EVALUACIONES / TAREAS. Revise que el trabajo enviado sea el correcto y a este destino.
- Requisitos para revisar el programa:
    - Las funciones deben tener: los nombres indicados en cada ejercicio, el número de cada ejercicio y deben estar en la solución ordenados de acuerdo a su número de ejercicio.
    - Buenas prácticas de programación:
        - Las funciones deben tener documentación interna (COMENTARIOS EN EL PROGRAMA FUENTE), al menos: lo que hace, las entradas y las salidas. También ponga documentación interna en aquellas partes que ameriten una explicación adicional para un mejor entendimiento del fuente.
        - Nombres significativos de las variables.
- Una ventaja de las funciones es su reutilización, es decir, el uso de funciones que han sido desarrolladas previamente. En esta tarea se introduce el concepto.
- Cada ejercicio vale 10 puntos.
- Fecha de entrega: 8 de marzo 11:30pm.
- Se coordinará día y hora para revisar la evaluación junto con el estudiante quien siendo el autor mostrará el dominio de la solución implementada desde el punto de vista técnico (uso de conceptos de programación y del lenguaje) así como de la funcionalidad (lo que hace la solución). La revisión puede constar de las siguientes actividades:
    - Revisar esta solución particular
    - Revisar conceptos incluidos en la evaluación
    - Aplicar otras actividades con una complejidad igual o menor a la evaluación.

En esta práctica se va a introducir la reutilización de software propio: desarrollar software, en este caso funciones, utilizando funciones que usted ha desarrollado previamente.

SUGERENCIA: Siga la metodología de solución de problemas: entender el problema, diseñar un algoritmo, codificar y probar programa. En la etapa de diseño del algoritmo haga un esquema para determinar el comportamiento o patrón del algoritmo, luego proceda con su desarrollo.
NO SE VAYA DIRECTAMENTE A PROGRAMAR EN LA COMPUTADORA, PRIMERO PLANIFIQUE LA SOLUCIÓN HACIENDO ESE ESQUEMA Y LUEGO HAGA EL PROGRAMA.

Por favor compartan conocimientos e ideas. Para ello pueden trabajar en grupos de 3 personas, pero la presentación y evaluación del trabajo es individual. Es decir que cada miembro del grupo debe enviar el trabajo y su nota será de acuerdo a la revisión particular. Si trabaja en grupos escriba en el fuente su nombre como autor del trabajo y los nombres de los compañeros como coautores.

Ejercicios:

1) Se requiere la función calificacion para pasar de una escala numérica a alfabética. La función va a recibir una calificación que es un número natural entre 0 y 100, de lo contrario retorna el string “ERROR: NOTA DEBE ESTAR ENTRE 0 Y 100”.

Debe retornar los siguientes valores:
“A – Excelente (Aprobado)”: notas de 90 a 100
“B – Bien (Aprobado)”: notas de 80 a 89
“C – Suficiente (Aprobado)”: notas de 70 a 79
“D – Deficiente (Reprobado)”: notas de 50 a 69
“F - Muy deficiente (Reprobado)”: notas de 0 a 49

Ejemplos del funcionamiento:

```
>>> calificacion(95)
A – Excelente (Aprobado)

>>> calificacion(65)
D – Deficiente (Reprobado)

>>> calificacion(-25)
ERROR: NOTA DEBE ESTAR ENTRE 0 Y 100
```

2) Haga la función pares_impares que reciba un número natural de 4 dígitos y retorne dos valores: el primero va a contener todos los dígitos pares y el segundo todos los dígitos impares que aparecen en el número de entrada. Cuando no hayan dígitos pares o impares debe retornar el valor -1 en la parte respectiva del resultado. El orden de los dígitos en los resultados debe ser acorde al orden en el número de entrada. Validar que el número de entrada esté en el rango indicado, de lo contrario retornar el mensaje “ERROR: DEBE SER UN NÚMERO NATURAL DE 4 DÍGITOS”. Ejemplos del funcionamiento:

```
>>> pares_impares(7851)
(8, 751)

>>> pares_impares(1234)
(24, 13)

>>> pares_impares(2426)
(2426, -1)

>>> pares_impares(3557)
(-1, 3557)

>>> pares_impares(219999)
ERROR: DEBE SER UN NÚMERO NATURAL DE 4 DÍGITOS
```

3) Una función boolena es una función que retorna solamente 2 valores: True o False. Haga la función booleana bisiesto que reciba un año como parámetro (número natural de 4 dígitos >= 1800) y retorne el valor booleano de verdadero (True) si el año es bisiesto o el valor booleano de falso (False) si el año no es bisiesto. Investigue las condiciones para que un año sea bisiesto. En caso de que no se cumpla con la restricción del año retornar el valor False.

   Ejemplos del funcionamiento:

```
>>> bisiesto(2016)
True

>>> bisiesto(2000)
True

>>> bisiesto(2100)
False

>>> bisiesto(2013)
False

>>> bisiesto(1650)
False
```

4) Haga la función booleana valida_fecha para determinar si una fecha esta correcta o incorrecta. La función recibe un entero positivo de 8 dígitos en el formato ddmmaaaa: dd son los días, mm son los meses y aaaa es el año. Una fecha se considera correcta si cumple con estas condiciones:

- el año debe ser >= 1800
- el mes debe estar entre 1 y 12
- el día debe ser según el mes y el año. Meses con 31 días: enero, marzo, mayo, julio, agosto, octubre y diciembre. Meses con 30 días: abril, junio, setiembre y noviembre. Febrero tiene 28 días cuando el año no es bisiesto, si el año es bisiesto febrero tiene 29 días. Reutilice software: para determinar si el año es bisiesto use la función bisiesto desarrollada anteriormente.

Si la fecha esta correcta retorna True de lo contrario retorna False.

Ejemplos del funcionamiento:

```
>>> valida_fecha(30032015)
True

>>> valida_fecha(31092006)
False

>>> valida_fecha(29022015)
False
```

5) La paridad de un número entero se refiere a su atributo de ser par o impar. La paridad par se refiere a que el número es divisible por 2, y la paridad impar se refiere a que no es divisible por 2. Comparativamente dos números pueden tener una misma paridad. Haga la función paridad que reciba dos números enteros y retorne “Paridad Par” si ambos tienen paridad par, o “Paridad Impar” si ambos tienen paridad impar y “DIFERENTE PARIDAD” si tienen diferente paridad. Ejemplos del funcionamiento:

```
>>> paridad(2, 4)
Paridad Par

>>> paridad(11, 9)
Paridad Impar

>>> paridad(15, 20)
DIFERENTE PARIDAD

>>> paridad(11, -21)
Paridad Impar
```

En este ejercicio no valide los datos, siempre va a recibir números enteros.

6) Haga una función llamada doble_de_iImpar que reciba un número natural y retorne True si éste es el doble de un número impar, de lo contrario retorne False. Ejemplos:

```
>>> doble_de_impar (21)
False

>>> doble_de_impar(14)
True

>>> doble_de_impar(8)
False

>>> doble_de_impar(15)
False
```

7) Manejo de una cuenta bancaria de ahorros con la función cuenta_bancaria. La función va a recibir 3 argumentos: el saldo actual de la cuenta (>=0), el tipo de operación que el usuario quiere hacer (el tipo puede ser el valor 1 que significa una operación de depósito, o un 2 que significa una operación de retiro), la cantidad de la operación (>0). Debe retornar el nuevo saldo. Tanto los depósitos como los retiros deben ser múltiplos de 1000. Hay que validar los datos de entrada según se indicó anteriormente y revisar en los retiros que estos se puedan hacer en caso de que el saldo lo permita, de lo contrario retornar el valor booleano False.

   Ejemplos del funcionamiento:

```
>>> cuenta_bancaria(25000, 2, 5000)
20000

>>> cuenta_bancaria(25000, 1, 3000)
28000

>>> cuenta_bancaria(28000, 2, 1200)
False                → la cantidad de la operación no es múltiplo de 1000
```

8) Haga la función pago_celular para calcular y retornar el monto a pagar por servicios de telefonía celular. Recibe estos argumentos para calcular este monto: la cantidad de minutos consumidos en llamadas (entero >=0), la cantidad de mensajes enviados (entero >=0) y el tipo de plan de uso de Internet (1 dígito) . Use la siguiente tabla escalonada para calcular el monto a pagar:

- Tarifa básica de 2750 colones, dando derecho a 60 minutos de consumo. Si usa menos minutos debe pagar esta tarifa mínima.
- Si consume más de 60 minutos y menos de 121, paga la base más 50 colones adicionales por cada minuto en ese rango.
- Si consume más de 120, paga la base más 60 minutos a 50 colones más 35 colones adicionales por cada uno de los minutos adicionales a 120
- Al costo de las llamadas se agrega el costo de cada mensaje: 3 colones.
- También se agregar el costo de uso de Internet de la siguiente manera:

Si el tipo de plan es 0 no hay uso de Internet.
Si el tipo de plan es 1 se cobra 12000 colones.
SI el tipo de plan es 2 se cobra 15000 colones.
Si el tipo de plan es 3 se cobra 25000 colones.
Otros valores en el tipo de plan no son permitidos.

Adicionalmente debe agregarle al monto a pagar un impuesto de ventas del 13%. Luego agregue una colaboración de 200 colones para el Servicio de la Cruz Roja.
Haga las validaciones de las restricciones y si encuentra algún error retorne el mensaje respectivo.

9) Haga la función nombre_dia que reciba una fecha como un entero de 8 dígitos estructurados así ddmmaaaa: los dos primeros dígitos de la izquierda representan el día, los siguientes dos dígitos son el mes y los cuatro dígitos de la derecha son el año. Debe retornar el nombre del día correspondiente a esa fecha. Sugerencia: estudie algoritmo de Zeller. Valide que la fecha sea correcta, de lo contrario retorne False. Ejemplo del funcionamiento:

```
>>> nombre_dia(24022016)
Miércoles

>>> nombre_dia(29022023)
False
```

10) Haga una función booleana para determinar si un número es palíndromo. La función recibe un número natural de 4 dígitos y retorna True si el número es palíndromo y False de lo contrario. Un número es palíndromo cuando se lee igual de izquierda a derecha y viceversa. No valide los datos de entrada, llegan según especificaciones. Para este ejercicio vamos a reutilizar la función al_reves desarrollada en trabajos anteriores: ponga el fuente de dicha función en este programa fuente de la tarea y luego la llama en la nueva función para darle vuelta al número de entrada. Si el valor retornado por la función al_reves es igual al número de entrada entonces el número es palíndromo. Ejemplos del funcionamiento:

```
>>> es_palindromo(1441)
True

>>> es_palindromo(1234)
False          → se lee 1234 de izquierda a derecha, se lee 4321 de derecha a izquierda
```

Última línea.