# Primer Parcial de Programación 1

25/09/2019

**Tener en cuenta que:**

- La duración del parcial es de 3 horas.
- Se debe realizar en computadora, para este fin se tendrá que crear una carpeta primerparcial2019 y colocar dentro las clases solicitadas en el ejercicio 2. Se recomienda utilizar la unidad Z para crear esta carpeta.
- Culminado las horas del parcial se deberá esperar al docente para realizar la copia del trabajo realizado.
- Tenga en cuenta que no se tomaran en cuenta para la corrección clases que no compilen.
- No se puede usar material, ni uso de internet, ni celulares durante la realización del parcial.
- Se contestan dudas solo en la primer hora y media del parcial.

## Ejercicio 1- Teórico (Tiempo recomendado 30 minutos)

a) Dado los siguientes números convertir:
   a. 45 a base 2
   b. $10001_2$ a base 8
   c. $A1_{16}$ a base 2

b) Explique el funcionamiento de la función range brindada por Python y como podemos usarla en un bloque for para iterar sobre una lista.

## Ejercicio 2 - Practico (Tiempo recomendado 2h)

Una compañía proveedora de energía eléctrica busca optimizar el consumo de energía de sus clientes, para ello diseño un plan de tarifas que busca premiar a aquellos clientes que usan energía fuera del horario pico. La energía usada de una residencia se mide en kWh (kilovatio hora) a través de un dispositivo llamado contador. Estos aparatos son dispositivos digitales que contabilizan por hora el consumo que tiene la residencia. Internamente por día mantienen 24 registros de la cantidad de kWh que fueron consumidos en cada hora del día.

El plan de tarifas propuesta considera 3 opciones:

- **Plan residencial simple**
  - En este plan se tiene una única franja de horarios a un mismo precio.
  - Las tarifas vigentes en este plan son:
    - Horario completo (00 a 23:59) : $ 6,47 x kWh.
- **Plan residencial doble horario**
  - Se definen dos franjas de horarios, en lo que se beneficia el uso de energía fuera del horario pico con un valor más económico.
  - Las tarifas vigentes en este plan son:
    - Horario Fuera de Punta (23:00 a 17:00 hrs.) : $ 3,453 x kWh
    - Horario Punta (17:00 a 23:00hrs): $ 8,623 x kWh
- **Plan residencial triple horario**
  - Se definen tres franjas de horarios, en lo que se beneficia el uso de energía fuera del horario pico, pero además se da un beneficio extra si se usa energía en el horario de menor consumo.
  - Las tarifas vigentes en este plan son:
    - Horario Valle (00:00 a 07:00 hrs.): $1,803 x kWh
    - Horario Llano (07:00 a 17:00 y 23:00 a 00:00 hrs.): $4,676 x kWh
    - Horario Punta (17:00 a 23:00 hrs.): $8,623 x kWh

Con esto en mente se debe realizar un conjunto de operaciones que permitan asistir a la empresa de energía a gestionar los gastos de una residencia. Cada día de consumo se presenta como un vector de 24 registros que indica el consumo que obtuvo en cada hora del día.

Se debe generar un módulo de nombre primer_parcial.py que contenga las siguientes operaciones:

- def **calculador_consumo_dia_franja**(consumo_dia_esp, hora_inicio, hora_fin)
  - **Función que retorna** el total de kWh consumidos en un día especifico en una franja de horarios.
  - Recibe como argumento:
    - **consumo_dia_esp** que es una lista de 24 posiciones con el consumo reportado por hora de un contador para un día. En la posición 0 de la lista se tiene el consumo en kWh desde las 0 hasta las 0:59, en la posición 23 se tiene el consumo en kWh desde la 23 hasta las 23:59.
  - Por ejemplo podría llamarse a esta función de la siguiente forma calculador_consumo_dia_franja([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 10, **15**, **10**, **10**, 1, 1], 19, 22), esto debería brindar el consumo de las 19:00 hasta las 22:00 = 15 (19 a 20) + 10 (20 a 21) + 10 (21 a 22) = 35
- def **calcular_gasto_dia**(consumo_dia_esp, tipo_tarifa)
  - **Función que retorna** el gasto en pesos de una residencia para un día de consumo.
  - Recibe como argumento:
    - **consumo_dia_esp** que es una lista de 24 posiciones con el consumo reportado por hora de un contador para un día.
    - **tipo_tarifa** contiene el tipo de tarifa usada para realizar el cálculo, siendo los valores posibles 1 – residencial simple, 2 – residencial doble horario, 3 – residencial triple horario.
  - Se recomienda utilizar la función anterior para determinar el consumo x franja.
- def **calcular_gastos_mensual**(consumo_mensual, tipo_tarifa)
  - Función que retorna el gasto en pesos de una residencia para un mes de consumo.
  - Recibe como argumento:
    - **consumo_mensual** es una matriz que contiene 31 filas representando los días de un mes y 24 columnas representando el consumo x hora (de la misma manera que se recibe en las dos operaciones anteriores). Por ejemplo, en la posición [0][5] de la matriz se representa el consumo del día 1, de las 5:00 a las 6:00.
    - **tipo_tarifa** contiene el tipo de tarifa usada para realizar el cálculo, siendo los valores posibles 1 – residencial simple, 2 – residencial doble horario, 3 – residencial triple horario.
- def **franja_horario_mayor_consumo_mensual**(consumo_mensual)
  - Función que retorna la franja de horarios en el mes en la cual se consumió mayor cantidad de kWh. Para determinar las franjas se tiene que considerar segmentos de 4 horas comenzando de las 0 hasta las 23:00. El resultado se retorna en una lista de dos valores, donde el primer valor indica el horario de comienzo de la franja y el segundo el horario de fin. Un resultado posible es de [19, 22] indicando que el mayor consumo se obtuvo comenzando a las 19 y terminando a las 22:59.
  - Recibe como argumento:
    - **consumo_mensual** es una matriz que contiene 31 filas x 24 columnas representando consumo por día x hora (ver detalle operación anterior).
  - Se recomienda primero calcular una lista con el acumulado mensual por hora (24 posiciones donde cada posición contiene el acumulado x hora en kWh del mes). Y a partir de esa lista calcular la franja de mayor consumo.

Al final del módulo se debe colocar un conjunto de sentencias que prueben el correcto funcionamiento de cada operación. Para facilitar el trabajo se brinda un esqueleto del módulo con un set de pruebas inicial, se debe agregar al menos una prueba más de cada operación.