PARCIAL PYTHON

LIC. NATALIA COLUSSI PROF. DAMIÁN MAROTTE

Redictado Programación II 27 de Abril 2022

Consideraciones Generales

Antes de comenzar a resolver el examen tener en cuenta lo siguiente:

(a) Las funciones deben contener su receta completa.

(b) El testing se realizará mediante la librería pytest, se recomienda no menos de tres casos de test por función.

(c) Guardar la solución de cada ejercicio en un archivo diferente, cuyo nombre sea por ejemplo: **ejercicio1-PerezAna.py**, es decir, contenga el numero de ejercicio y el nombre de ustedes. Dentro del archivo escriban un encabezado también con su nombre y apellido junto con el legajo o DNI.

(d) El ejercicio 4 se entrega en formato papel corresponde al ejercicio de teoría de los temas evaluados.

Ejercicio 1: Clave Maestra

Atención:*Use ciclos definidos para recorrer la String en este problema*.

La cerradura de una caja fuerte posee doble puerta de seguridad y con ello dos claves por separado: una clave primaria y una secundaria. La representación de la información se hace de una manera compacta utilizando una *string* como número único de dato de acceso a la caja fuerte. Así:

- los números que están ubicados en las **posiciones pares** conforman la clave primaria, que permite acceder a *primera puerta* de la caja fuerte. Por ejemplo, si la clave maestra es el dato: “47582” , entonces la clave primaria será:“452”
- los números que están ubicados en las **posiciones impares** conforman la clave secundaria, que permite acceder a *segunda puerta* de la caja fuerte. Por ejemplo, si la clave maestra es el dato: “47582” , entonces la clave secundaria será:“78”

Se pide entonces el diseño de las siguientes tres funciones:

(a) Diseñar la función: **convertir_claveMaestra: String String** $\rightarrow$ **String** la cual dadas las claves primaria y secundaria genera una clave maestra.

(b) Diseñar la función: **extraer_clavePrimaria: String** $\rightarrow$ **String** la cual dada la clave maestra extrae la clave primaria.

(c) Diseñar la función: **extraer_claveSecundaria: String** $\rightarrow$ **String** la cual dada la clave maestra extrae la clave secundaria.

```
convertir_claveMaestra ("758","841") ="785481"
extraer_clavePrimaria ("785481") = "758"
extraer_claveSecundaria ("785481") ="841"
```

Ejercicio 2: Tupla Empleado

Atención: *Use modularización en la resolución de este problema*.

(a) Diseñe el tipo de dato: ***Empleado*** el cual contiene información sobre: *nombre, antigüedad, salario bruto, descuentos,* y *bonificaciones* de un empleado de la industria ACME.

(b) Diseñe una función que construya una tupla ***Empleado*** válida, llamada **empleado_valido** teniendo en cuenta las siguientes restricciones que rigen en la industria ACME:

- El salario bruto inicial es de $55,000 salvo que se indique lo contrario con un valor de argumento diferente.
- La antigüedad inicial es 0 salvo que se indique lo contrario.
- Los restantes datos: *descuentos* y *bonificaciones* se inicializan en cero.

```
empleado_valido ("Frida",5, 75000) = ("Frida", 5, 75000,0,0)
empleado_valido ("Fido",3) = ("Fido", 3, 55000,0,0)
empleado_valido ("Francis") = ("Francis",0, 55000,0,0)
```

(c) Diseñe una función que visualice la información de la tupla **Empleado** de una manera organizada para que rápidamente un usuario pueda visualizar la información asociada al mismo. Llamar a esta función **mostrar_empleado**.

(d) Diseñe la función **calcula_sueldo** la cual recibe un **Empleado** y lo devuelve actualizado en base a la siguientes reglas y criterios contables.

- Los descuentos se realizan sobre el sueldo en bruto y corresponden al %8 del mismo.
- La bonificación aplica las siguientes condiciones según su antigüedad:
  (a) Si lleva más de 11 años en la empresa se le aplica un aumento del 10 %.
  (b) Si lleva menos de 10 años pero más que 5 se le aplica un aumento del 7 %.
  (c) Si lleva menos de 5 años pero más que 3 se le aplica un aumento del 5 %.

(d) Si lleva entre 1 y 3 años se le aplica un aumento del 3 %.

- El cálculo del sueldo en neto se puede obtener a partir del sueldo en bruto, los descuentos y las bonificaciones.

Ejercicio 3: Recursión

Los números de **tribonacci** son parecidos a los números de *Fibonacci*, pero en lugar de comenzar con dos valores básicos predeterminados, se dispone inicialmente de tres valores básicos predeterminados:

$$\mathrm{tribonacci}(0) = 0 \quad \mathrm{tribonacci}(1) = 0 \quad \mathrm{tribonacci}(2) = 1.$$

La secuencia comienza con estos tres valores y cada termino luego de éstos es el resultado de sumar los tres valores precedentes.

$$\mathrm{tribonacci}(n+3) = \mathrm{tribonacci}(n+2) + \mathrm{tribonacci}(n+1) + \mathrm{tribonacci}(n) \text{ con } n >= 0$$

Algunos números de esta secuencia son:

$$0, 0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, 274, 504, 927, 1705, 3136, 5768, 10609 \ldots$$

Te pedimos que diseñes de forma recursiva la función de **tribonacci**.

Ejercicio 4: [TEORÍA] Ciclos en Problemas

Nuestra amiga, Ana Lahacker, nos avisó que los ciclos que te mostraremos mas abajo, estaban mal diseñados. Te pedimos que la ayudes analizando los mismos para así justificar su afirmación y respondas a las siguientes preguntas:

(a) Elija uno de ellos y construya una tabla de iteraciones.
(b) ¿Qué tipo de ciclo son? Clasifíquelos.
(c) ¿Cuál es el problema que tienen?. Indíquelo para cada uno de ellos.
(d) Proponga una solución para corregirlos.

a) Ciclo 1

```
1 n = 1
2 while True:
3     if (n % 2 ==0) and (n % 2 !=0):
4         break
5     n=n+1
```

b) Ciclo 2

```
1
2 count = 0
3 while True:
4     print (count)
5     count += 1
6     if count >= 5:
7         continue
```

c) Ciclo 3

```
1     maxVal= int (input ("Ingrese un entero positivo"))
2     i=0
3     while(i>= maxVal):
4         i=i+1
5     print("El valor de i es:", i)
```