Argentina Programa 4.0

Universidad Nacional de San Luis

Desarrollador Python

Lenguaje de Programación Python: Soporte Funcional

Autor:

Dr. Mario Marcelo Berón

Argentina Programa 4.0
Universidad Nacional de San Luis

Práctico Nro. 7.2: *Llamables*

Ejercicio 1: Defina la función sumaTupla la cual recibe una tupla de números enteros como prámetro y retorna como resultado la suma de dichos números.

Ejercicio 2: Defina la función minMax la cual recibe como parámetro una tupla de números enteros y retorna como resultado la suma del número mínimo con el número máximo.

Ejercicio 3: Defina la función todosNúmeros la cual recibe una tupla y retorna como resultado True si todos los ítems de la tupla son números enteros y False en otro caso.

Ejercicio 4: Defina la función tipoDominante la cual recibe una lista de ítems de los típos básicos de Python (bool, int, float, string) y retorna como resultado 0 si el tipo dominante es bool, 1 si el tipo dominante es int, 2 si el tipo dominante es float y 3 si el tipo dominante es string. En cualquier otro caso la función genera una excepción TypeError.

Ejercicio 5: Defina la función first la cual recibe como parámetro una lista y retorna como resultado el primer elemento de la lista.

Ejercicio 6: Defina la función rest la cual recibe como parámetro una lista y retorna como resultado todos los elementos de la lista con excepción del primero.

Ejercicio 7: Realice las funciones de los ejercicios 5 y 6 pero con tuplas.

Ejercicio 8: Defina la función lista la cual recibe un número variable de ítems de tipos básicos y retorna como resultado una lista.

Ejercicio 9: Defina la función nthcdr la cual recibe como parámetro un entero n y una lista l y retorna como resultado la lista pero con los n primeros ítems elminados.

Ejercicio 10: Defina la función butlast la cual recibe como parámetro un entero n y una lista l y retorna la lista pero con los n últimos ítems eliminados.

Ejercicio 11: Defina la función mapcar la cual recibe como parámetro una función de un argumento f y una lista l. La función retorna como resultado una lista cuyos ítems resultan de la aplicación de f a cada uno de los elementos de la lista l.

Ejercicio 12: Defina la función remove-if la cual recibe como parámetro un predicado p y una lista l y retorna como resultado una lista que contiene los elementos de l excepto aquellos elementos para los cuales el predicado p es verdadero.

Ejercicio 13: Defina la función apply la cual recibe como parámetro una función f y una lista l. La función retorna como resultado la aplciación de f con l como argumento.

Ejercicio 14: Defina una función lambda que permita determinar si un número es par.

Ejercicio 15: Defina una función lambda que reciba como parámetro dos números enteros y retorne como resultado el mayor o, en caso de igualdad, el primer argumento.

Ejercicio 16: Defina la función ordenar la cual recibe como parámetro una lista de duplas l. Las duplas contienen números enteros. La función retorna como resultado la lista l ordenada por la primer componente de cada dupla.

Ejercicio 17: Defina funciones lambda para:

1. Calcular el doble de un número.

2. Determinar si un número es impar.
3. Dados dos strings retornar el de mayor longitud.
4. Dada una dupla retornar otra cuya primera componente sea el doble de la primer componente de la dupla de entrada y el segundo ítem sea el triple del segundo ítem de la dupla de entrada.
5. Determinar si un número es mayor que 0.
6. Determinar si un número está dentro de un rango determinado.
7. Determinar si un punto está dentro de una circunsferencia.
8. Calcular el área de un triángulo.
9. Calcular el área de un cuadrado.
10. Ordenar de forma ascendente o descendente una lista de números enteros.

Ejercicio 18: Escriba una función que retorne como resultado otra función la cual permite pasar un string a mayúsculas.

Ejercicio 19: Escriba una función que retorne como resultado otra función la cual permite pasar un string a minúsculas.

Ejercicio 20: Escriba un programa que permita ver el uso de las funciones definidas en los ejercicios 18 y 19.

Ejercicio 21: Defina una función que permita generar la función definida en el ejercio 18 o en el ejercicio 19 dependiendo del valor de un parámetro.

Ejercicio 22: Escriba un programa que permita ver el uso de la función definida en el ejercicio anterior.

Ejercicio 23: Defina la clase Triángulo que permita agrupar las funciones: área, perímetro e hipotenusa.

Ejercicio 24: Defina la función generarÁrea la cual tiene tres parámetros el primero indica qué área quiere generar si la de un triángulo o la

de un rectángulo. Los parámetros restantes son la base y la altura. A continuación se muestran ejemplos de uso de la función:

```
t1=generarÁrea(True,2,3)
r1=generarÁrea(True,2,2)
print(" Area del Triángulo:",t1)
print(" Area del Rectangulo:",r1)
```

Ejercicio 25: Defina la función listaDeÁreas la cual recibe como parámetro un entero n y retorna como resultado una lista de funciones construida de la siguiente manera: Cuando n es par se genera el área de un triángulo cuando n es impar se genera el área de un rectángulo.

Ejercicio 26: Defina la función invocarFunciones la cual recibe como parámetro una lista de funciones y generadas por listaDeÁreas y las invoca.

Ejercicio 27: Defina la clase Par la cual permite registrar un número par e implementa un iterador cuya función es obtener el siguiente par. Por ejemplo si se crea un objeto p=Par(2) y se invoca a next(p) el resultado es 4, si se vuelve a invocar next(p) el resultado es 6 y así sucesivamente.

Ejercicio 28: Defina la clase Cadena la cual permite almacenar una lista de cadena de caracteres. La clase implementa un iterador que retrona las cadenas de la lista invertidas.

Ejercicio 29: Defina la clase ContadorMódulo la cual implementa un contador módulo un número establecido por el usuario. Ejemplo:

```
c=ContadorMódulo(3)
i=iter(c)
i.next() # 0
i.next() # 1
i.next() # 2
i.next() # 0
```