PRELIMINARES

Una función es un conjunto de instrucciones que se ejecutan cada vez que se "llama"/"invoca" a la función desde el programa. La función más usada en Python es la función print(), que permite desplegar información por pantalla. La segunda función más usada es input(),que permite capturar información desde la pantalla (y también desplegar mensajes si se escribe un texto entre comillas entre los paréntesis redondos). La tercera función más usada es range(),que hemos usado en los ciclos for.

En un programa se puede llamar a funciones ya creadas por otros (como print()) o funciones creadas por un@ mism@.

En el primer caso (llamar a funciones ya creadas por otros), para poder invocar a la función dentro del código es necesario incluir la biblioteca (módulo) en la que esta función se encuentra declarada con el comando import. Por ejemplo, la función print() se encuentra declarada en el módulo **builtins (import builtins)**. Sin embargo, no necesitamos escribir import **builtins** porque esa opción viene por omisión. Para llamar funciones de cualquier otra biblioteca, se debe escribir primero el nombre del módulo, seguido de un punto y luego el nombre de la función. Por ejemplo: **builtins.print("hola mundo")**. Veamos como quedaría el código para mostrar el mensaje "hola mundo", realizando la importación de la biblioteca builtins y el llamado a la función print():

```
import builtins
builtins.print("hola mundo")
```

Como se puede observar:

- en la primera línea se importa la biblioteca **builtins (y así tenemos acceso a todas las funciones definidas en dicha biblioteca)**
- en la segunda línea, llamamos a la función print correspondiente a builtins para poder desplegar el mensaje hola mundo por pantalla.

Una de las bibliotecas más comunes en Python es math. Dicha biblioteca da acceso a múltiples funciones matemáticas, entre las que se encuentran:

- ceil(x): retorna el entero más pequeño que sea igual o mayor que *x*
- exp(x): retorna el valor de e^x
- fabs(x): retorna el valor absoluto de x
- factorial(x): retorna el factorial de x
- floor(x): retorna el entero más grande que sea igual o menor que *x*
- log(x): retorna el logaritmo de x
- log(x,baseLog): retorna el logaritmo de x en base baseLog
- pow(x,y): retorna x elevado y
- sqrt(x): retorna la raiz cuadrada de x

Para listar todas las funciones de un módulo importado, escribe help(nombreModulo).

```
import math
help(math)
```

En el segundo caso (para llamar a funciones creadas por un@ mism@), es necesario definir la función. Esto es:

- definir el nombre de la función
- definir sus eventuales argumentos de entrada
- escribir el código que debe ejecutarse cuando se invoca a la función
- retornar de la función

Nombre de la función: Este es el nombre con el que se llama a la función. Puedes usar cualquier palabra (o combinación de palabras) no reservada del lenguaje Python. Para indicar que es una función, Python utiliza la palabra reservada def antes del nombre de la función. Por ejemplo, si queremos crear una función cuyo nombre sea calculo, tendremos que colocar def calculo( ):

Argumentos (o parámetros) de entrada: En Matemáticas, hay funciones que reciben un valor de entrada (por ejemplo, la función $f(x)=x^2$ necesita conocer el valor de x para poder indicar el valor que retorna la función) y funciones que no (por ejemplo, y=1). En Computación también hay funciones que reciben y que no reciben valores de entrada. Cuando una función recibe valores de entrada (valores que necesita para poder realizar los cálculos para los que fue programada), puedes pensar en cada argumento de entrada como una "caja" donde la función guarda el valor que recibe. En este caso, para cada argumento de entrada se debe indicar el nombre de la "caja" que se usará dentro de la función para procesar el dato de entrada. Por ejemplo: def suma(a, b): es el nombre de la función suma que recibe los valores de dos parámetros y los almacena en las variables a y b.
Cuando se trata de una función que no requiere parámetros de entrada, ya sea porque los datos que requieren se los solicita directamente al usuario, en vez de listar los parámetros de entrada se deja un espacio en blanco. Por ejemplo: si se desea definir una función, llamada suma_100, que sume los primeros 100 números naturales, la definimos como def suma_100().

Implementación de la función: En el cuerpo de la función (en la línea inmediatamente debajo del nombre de la función e indentado a la derecha) se escriben todas las instrucciones que se desean ejecutar cada vez que la función se llame desde el programa principal.

Valor de retorno de la función: Tal como en Matemáticas, hay funciones que retornan un resultado. Por ejemplo, la función $f(x)=x^2$. Cada vez que se ingresa un valor de x, la función $f(x)=x^2$ "retorna" el valor de x al cuadrado. En Computación, hay funciones que retornan un resultado **pero también hay funciones que no retornan un resultado** (el segundo caso resulta muy extraño para los principiantes porque sus conocimientos de Matemáticas les indican que todas las funciones retornan un resultado). Las funciones que retornan resultados, al igual que las funciones matemáticas, son aquellas que deben entregar de vuelta un valor que se calcula dentro de la función. Para retornar el valor de la función, se utiliza el comando return seguido del valor a retornar. Por ejemplo, si programas una función donde la variable **`total`** calcule la suma de los primeros 10 números enteros, y se quiere retornar este valor, entonces es necesario utilizar la línea de comando return **total**. Si la función programada no retorna un resultado (por ejemplo, una función que solo imprime el contenido de un arreglo por pantalla), se utiliza el mismo comando seguido de ningún otro nombre de variable o valor, es decir, simplemente se coloca return.

A modo de resumen, las funciones se pueden clasificar de dos maneras: si reciben o no parámetros de entrada y si retornan o no valores. El siguiente cuadro muestra en detalle cómo se escribe y cómo se podría utilizar cada una.

<table>
<tr><td></td><td></td><td colspan="2">Funciones que retornan valores</td></tr>
<tr><td></td><td></td><td>Si</td><td>No</td></tr>
<tr><td rowspan="2">Funciones que reciben parámetros</td><td>Si</td><td>Descripción genérica:
<pre>def <nombre>(<parametros>):
    <instrucciones>
    return <valor/variable></pre>Ejemplo de declaración:
<pre>def promedio(a,b):
    return (a+b)/2

Llamada desde programa:
prom = promedio(val1,val2)</pre></td><td>Descripción genérica:
<pre>def <nombre>(<parametros>):
    <instrucciones>
    return</pre>Ejemplo de declaración:
<pre>def promedio(a,b):
    print((a+b)/2)
    return

Llamada desde programa:
promedio(val1,val2)</pre></td></tr>
<tr><td>No</td><td>Descripción genérica:
<pre>def <nombre>():
    <instrucciones>
    return <valor/variable></pre>Ejemplo de declaración:
<pre>def promedio():
    a = int(input(“ingrese 1ra nota”))
    b = int(input(“ingrese 2da nota”))
    return (a+b)/2

Llamada desde programa:
prom = promedio( )</pre></td><td>Descripción genérica:
<pre>def <nombre>():
    <instrucciones>
    return</pre>Ejemplo de declaración:
<pre>def promedio():
    a = int(input(“ingrese 1ra nota”))
    b = int(input(“ingrese 2da nota”))
    print((a+b)/2)
    return

Llamada desde programa:
promedio( )</pre></td></tr>
</table>

Como puedes observar en los ejemplos, tal como en los condicionales y ciclos, después de la definición de la función está el carácter dos puntos (:). Además, se incluye una tabulación para definir que el código está dentro de la función que se esta definiendo.

Finalmente, la declaración de la(s) función(es) siempre se debe realizar al principio del programa, seguida(s) del código principal a ejecutar.

A continuación se presentan 3 ejemplos de códigos que cumplen con la misma funcionalidad (solicitar un número al usuario e informarle si el número es par o impar), pero que usan diferentes implementaciones de la función paridad(). Nota las diferencias entre los códigos cuando se usan funciones que retornan/no retornan un resultados y que piden/no piden argumentos de entrada.

Ejemplo 1: La función solicita al usuario un número por teclado (es decir, no toma argumentos de entrada) y despliega por pantalla el mensaje "Tu número es XX" (XX es la palabra "par" o "impar" según sea el caso).

```
def paridad(): #definición de la función
  num=int(input("escribe un numero: "))
  if(num%2==0):
     print("Tu numero es par ")
  else:
    print("Tu numero es impar ")
  return

paridad()
```

Ejemplo 2: La función recibe como argumento de entrada el número ingresado por el usuario y despliega por pantalla el mensaje "Tu número es XX" (XX es la palabra "par" o "impar" según sea el caso).

```
def paridad(numero): #definición de la función
  if(numero%2==0):
     print("Tu numero es par ")
  else:
     print("Tu numero es impar ")
  return

num=int(input("escribe un numero: "))
paridad(num)
```

Ejemplo 3: La función recibe como argumento de entrada el número del usuario y retorna un 0 si el número es par y 1 si es impar.

```
def paridad(numero): #definición de la función
  if(numero%2==0):
     return 1
  else:
     return 0

num=int(input("escribe un numero: "))
if(paridad(num)==1):
  print("Tu numero es par ")
else:
  print("Tu numero es impar ")
```

En el tercer ejemplo podemos ver que es posible retornar el valor en medio de una función, en este caso, el resto del código de la función no se ejecuta. Sin embargo, hay que tener cuidado, en caso que no se ejecuté ningún return, Python devolverá el valor None, el cual puede causar problemas en tu código.

PROBLEMAS PROPUESTOS

1. El siguiente código pide al usuario su año de nacimiento y le indica cuántos años cumple este año (2018).

```
def edad():
  print("¿En qué agno naciste?")
  anio=int(input())
  print("Este agno cumples ",2018-anio," años")
  return

edad()
```

a) Transforma el código para que mantenga la misma funcionalidad (solicitar al usuario su año de nacimiento e indicarle cuántos años cumple el año 2018) pero con una nueva función edad que recibe como argumento de entrada el año de nacimiento del usuario. La función no retorna valor alguno.

b) Transforma el código para que mantenga la misma funcionalidad (solicitar al usuario su año de nacimiento e indicarle cuántos años cumple el año 2018) pero con una nueva función edad que recibe como argumento de entrada el año de nacimiento del usuario y retorna el número de años que cumple el 2018.

Una solución posible (letra a):

```
def edad(anio):
  print("Este agno cumples ",2018-anio," años")
  return

print("¿En qué agno naciste?")
anioOrig=int(input())
edad(anioOrig)
```

Una solución posible (letra b):

```
def edad(anio):
  return 2018-anio

print("¿En qué agno naciste?")
anioOrig=int(input())
print("Este agno cumples ", edad(anioOrig)," años")
```

2. Para el siguiente programa, escribe la función pedir_dato() que solicita al usuario un número entero entre 1 y 100, ambos inclusive. La función debe validar que el número ingresado se encuentre en el rango. Si está en rango, retorna el valor del número ingresado. Sino, debe mantenerse pidiendo un número hasta que el usuario ingrese un valor válido.

Programa en Python a completar con la función:

```
#Escribe tu función aquí

valor=pedir_dato()
if(valor%2==0):
  print("Numero par")
else:
  print("Numero impar")
```

Una solución posible:

```
def pedir_dato():

  valor = int(input("Ingrese un valor entre 1 y 100: "))
  while(valor<1 or valor>100):
    valor = int(input("Ingrese un valor entre 1 y 100: "))
  return valor
```

3.

a) Escribe el código de la función tipo_triangulo() que recibe como argumentos de entrada las longitudes enteras de los 3 lados de un triángulo y retorna un 0 si el triangulo es escaleno, un 1 si es isósceles y un 2 si es equilátero.

b) Escribe un programa que le pida al usuario que ingrese la longitud de los lados de tantos triángulos como desee. El usuario debe indicar que ha finalizado el ingreso de datos ingresando un cero (0) en cualquiera de los campos. Al finalizar el ingreso de datos, el programa debe indicar el total de triángulos ingresados y cuántos triángulos de cada tipo fueron ingresados.

Una solución posible a)

```
def tipo_triangulo(a, b, c):
  if(a==b and b==c):
    return 0
  else:
    if(a==b or  b==c or a==c):
      return 1
    else:
      return 2
```

Una solución posible b)

```
def tipo_triangulo(a, b, c):
    if(a==b and b==c):
        return 0
    else:
        if(a==b or  b==c or a==c):
            return 1
        else:
            return 2
cont_esc=0  #contador escaleno
cont_iso=0  #contador isósceles
cont_eq=0   #contador equilátero
lado1=1
lado2=1
lado3=1
while (lado1!=0 and lado2!=0 and lado3!=0):
    lado1=int(input("Ingrese lado 1:"))
    lado2=int(input("Ingrese lado 2:"))
    lado3=int(input("Ingrese lado 3:"))
    if(lado1!=0 and lado2!=0 and lado3!=0):
        if(tipo_triangulo(lado1, lado2, lado3)==0):
            cont_eq=cont_eq+1
        else:
            if(tipo_triangulo(lado1, lado2, lado3)==1):
                cont_iso=cont_iso+1
            else:
                cont_esc=cont_esc+1

print("Número de triángulos ingresados: ", cont_eq+cont_esc+cont_iso);
print("Número de triángulos equiláteros: ", cont_eq);
print("Número de triángulos escalenos: ", cont_esc);
print("Número de triángulos isósceles: ", cont_iso);
```

4. Escriba una función que reciba como parametro dos números (num1 y num2) y muestre por pantalla todos los números entre num1 y num2 que son divisibles por 7, pero no por 5.

Una solución posible

```
def div7(num1, num2):
  for i in range(num1,num2+1,1):
    if ((i%7==0) and (i%5!=0)):
      print(i)
  return
```

5. Escriba una función que reciba como parametro dos números (num1 y num2) y muestre por pantalla una matriz de dos dimensiones, correspondiente a la multiplicación de los números.

Una solución posible

```
def mult(num1, num2):
  for i in range(1,num1+1,1):
    for j in range(1,num2+1,1):
      print(i*j,"\t",end="")
    print("")
  return
```

6. Escriba la función esPrimo que recibe un número y retornará 1 si el número es primo y 0 en caso contrario.

Una solución posible

```
def esPrimo(num):
  primo=1
  for i in range(2,num,1):
    if (num%i)==0:
      primo=0
      return primo
  return primo
```

7. Escriba un programa que le solicite un número al usuario e imprima por pantalla todos los números primos que dividen al número ingresado. Este programa debera hacer uso de la función definida en el punto 6.

Una solución posible

```
num = int(input("ingrese un numero: "))
for i in range(1,num+1,1):
    if ((num%i==0) and (esPrimo(i))):
        print(i)
```

8. Escriba una función que reciba dos puntos del espacio y retorne la distancia euclideana entre los puntos. Hint, utilice las funciones de la librería math
   pow(x,y): retorna x elevado y
   sqrt(x): retorna la raiz cuadrada de x

   Una solución posible

```
import math

def distEucl(X1,Y1,X2,Y2):
  return (math.sqrt(math.pow(X1-X2,2)+math.pow(Y1-Y2,2)))
```

9. Escriba una función que reciba dos números enteros y genere un número aleatorio entero entre estos dos números. Para ello importe la librería random y utilice la función random() que devuelve un número aleatorio entre 0 y 1.

   Una solución posible

```
import random

def numAleatorio(num1,num2):
   return int(random.random()*(num2+1-num1)+num1)
```

10. Cree una función que reciba un número entero y solicite al usuario adivinar el número ingresado. Para ello, la función debera solicitar por teclado un número al usuario. En caso que el número haya sido adivinado, entonces se desplegará por pantalla, que el número fue adivinado y retornara el valor 1. Caso contrario, la función retornará 0 y desplegará por pantalla la información, "el número ingresado es mayor que el número a adivinar" ó "el número ingresado es menor que el número a adivinar", según sea el caso.

    Una solución posible

```
def adivina(numAleat):
  num = int(input("ingresa un numero "))
  if (num==numAleat):
     return 1
  if (num<numAleat):
     print("el numero ingresado es menor que el numero a adivinar")
  else:
     print("el numero ingresado es mayor que el numero a adivinar")
  return 0
```

11. Cree un juego donde tenga que adivinar un número en 5 oportunidades, utilizando las funciones creadas en los ejercicios 9 y 10. En caso que lo adivine se desplegará felicitaciones, caso contrario se desplegará el número a adivinar.

Una solución posible

```
print("bienvenido al juego")
num1=int(input("ingrese el limite inferior "))
num2=int(input("ingrese el limite superior "))

numAdivinar=numAleatorio(num1,num2)
gano=0
contador=0
numTurnos=math.ceil(math.log2(num2-num1+1))

print("Tienes ",numTurnos, " turnos para adivinar, si sabes busqueda binaria, son suficientes intentos para adivinar")

while(gano==0 and contador<numTurnos):
   if adivina(numAdivinar)==0:
      contador=contador+1
   else:
      gano=1
if gano==0:
   print("perdiste el numero era ",numAdivinar)
else:
   print("FELICITACIONES GANASTE")
```

Esta solución determina el mejor número de opciones para adivinar, para obtener fijarlo en 5 cambiar la linea de código `numTurnos=math.ceil(math.log2(num2-num1+1))` por `numTurnos=5`