1000 PROGRAMADORES

# Introducción a la **Programación en Python**

Ministerio de Educación Cultura, Ciencia y Tecnología Gobierno de Salta

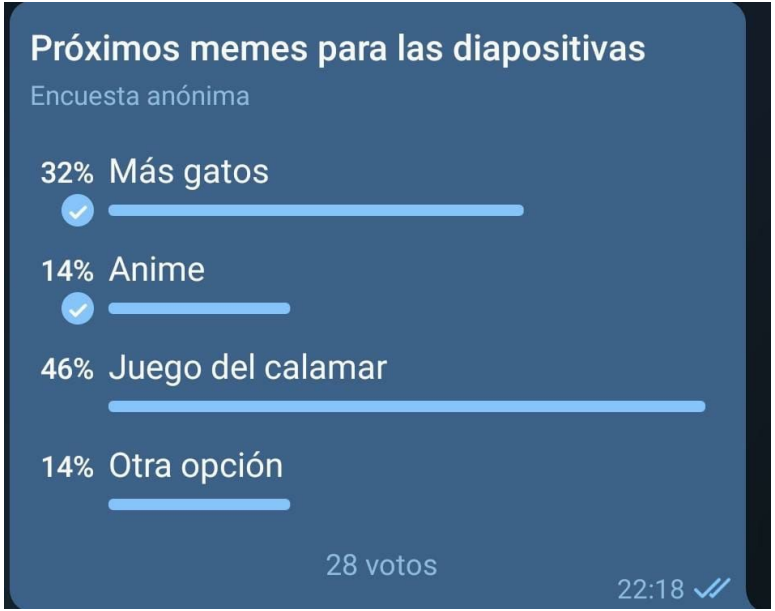
Ministerio de Economía y Servicios Públicos Gobierno de Salta

Universidad Nacional de Salta

# MÓDULO 5

**Args vs kwargs. Argumentos indeterminados.**
**Manejo de excepciones.**

Hoy presentamos la teoría con: El juego del Calamar

# ARGS - KWARGS

Python nos permite crear funciones que acepten un **número indefinido de parámetros** sin necesidad de que todos ellos aparezcan en la cabecera de la función.

Los operadores * y ** son los que se utilizan para esta funcionalidad.

# *args

***El parámetro especial *args en una función se usa para pasar, de forma opcional, un número variable de argumentos posicionales.***

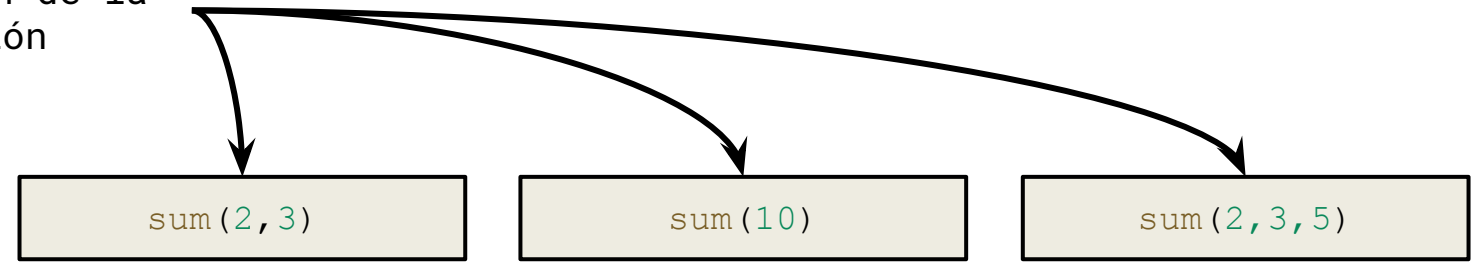
- ➔ *El símbolo '*' indica el tipo parámetro*
- ➔ *El nombre args se usa por convención.*
- ➔ *El parámetro recibe los argumentos como una tupla.*
- ➔ *Es un parámetro opcional.*
- ➔ *El número de argumentos al invocar a la función es variable.*
- ➔ *Son parámetros posicionales*

# Ejemplo:

Definición de la función

```
def sum(x, y):
    return x + y
```

Invocación de la función




5

error

error

# Función que imprime en pantalla

```
def varios_argumentos(*args):
 print(args)

varios_argumentos(1,2)
varios_argumentos(3,1,2,4)
varios_argumentos("hola")
```

# Función que suma números

```
def sumar(*numeros):
 suma=0
 for num in numeros:
   suma=suma+num
 return suma

print(sumar(1,2))
print(sumar(1,2,5))
print(sumar(10,20,30,40))
```

“Ayudemos” a los soldados del juego del calamar 😱

**Ejemplo:**

Los soldados tienen una lista con los números y otra lista con los nombres de los jugadores, y necesitan crear un diccionario con los siguientes datos: la clave será el nombre del jugador y el valor el número asignado a cada uno.
La idea es resolverlo usando *args

## Solución

```
def crear_diccionario(*datos):
 nombres=datos[0]
 numeros=datos[1]
 diccionario={}
 for pos in range(len(nombres)):
    nom=nombres[pos]
    num=numeros[pos]
    diccionario[nom]=num
 return diccionario

nombres=["Liu Kang","Peter","Chun Li"]
numeros=[1,200,5]
jugadores=crear_diccionario(nombres,numeros)
print(jugadores)
```

**Ejemplo:**

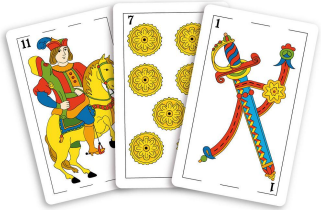
Diseñar un programa que genere aleatoriamente 3 cartas españolas (número y palo). Luego calcule y muestre el **ENVIDO** obtenido a través de esas cartas.

NOTA: *El puntaje del envido se calcula si se poseen dos o más cartas de igual palo, equivale a la suma del puntaje de las cartas del mismo palo más veinte puntos (10, 11 y 12 no suman).*

**31 puntos**

**26 puntos**

**31 puntos**



**7 puntos**

# Solución -> Funciones

```
import random
def validar_cartas(c1,c2,c3):
 res=0
 if(c1!=c2 and c1!=c3 and c2!=c3):
   res=1
 return res
```

```
def contar_envido(*cartas):
 if (len(cartas)==1):
   if (cartas[0]>=10):
     cuenta=0
   else:
     cuenta=cartas[0]
 else:
   cuenta = 20
   for n in cartas:
     if (n<10):
       cuenta = cuenta + n
 return cuenta
```

# Solución -> Script (parte 1)

```
palos=["oro","copa","basto","espadas"]
c1=[random.randrange(1,13),palos[random.randrange(0,4)]]
c2=[random.randrange(1,13),palos[random.randrange(0,4)]]
c3=[random.randrange(1,13),palos[random.randrange(0,4)]]
v=validar_cartas(c1,c2,c3)
print(c1,c2,c3)
```

¿Qué mejora puede hacerse? ¿En este juego, participan los cartas 8 y 9?

# Solución -> Script (parte 2)

```
if (v==1):
  if (c1[1]==c2[1] and c2[1]==c3[1]):
     env=contar_envido(c1[0],c2[0],c3[0])
  elif (c1[1]==c2[1]):
     env=contar_envido(c1[0],c2[0])
  elif (c1[1]==c3[1]):
     env=contar_envido(c1[0],c3[0])
  elif (c2[1]==c3[1]):
     env=contar_envido(c2[0],c3[0])
  else:
     env=max(contar_envido(c1[0]),contar_envido(c2[0]),contar_envido(c3[0]))
  print("Mi envido es: ",env)
else:
 print("Mazo trucado")
```

# **kwargs

***El parámetro especial **kwargs en una función se usa para pasar, de forma opcional, un número variable de argumentos con nombre.***

- ➔ *El símbolo '**' indica el tipo parámetro*
- ➔ *El nombre kwargs se usa por convención.*
- ➔ *El parámetro recibe los argumentos como un diccionario.*
- ➔ *Al tratarse de un diccionario, el orden de los parámetros no importa. Los parámetros se asocian en función de las claves del diccionario.*

# Ejemplo:

Definición de la función

```
def fun(**valores):
  print(valores)
```

Invocación de la función

```
fun(x=5, y=8)
```

```
{'x': 5, 'y': 8}
```

```
fun(x=5, y=8, z=4)
```

```
{'x': 5, 'y': 8, 'z': 4}
```

Ahora los soldados tienen otro problema más…

Los jugadores no pueden tener un número que no sea de tres cifras, el jugador 1 debería de tener el número “001”, respetando siempre el mismo formato como los otros jugadores.

```
def formatear_numero(**datos):
 nuevo_num=[]
 for jug,num in datos.items():
   numero_formateado=str(num)
   if (num<10):
     numero_formateado='00'+str(num)
   elif (num<100):
     numero_formateado='0'+str(num)
   nuevo_num.append(numero_formateado)
 return nuevo_num

nombres=["Liu Kang","Peter","Chun Li"]
numeros=[1,200,5]
numeros=formatear_numero(jug1=1,jug2=200,jug3=5)
```

Luego invocamos a la función **crear_diccionario** y conseguimos nuestro diccionario con el formato deseado


SOLDADO
YO
imgflip.com

**Ejemplo:**

Una sala maternal de la ciudad abrió sus inscripciones al ciclo lectivo 2022. Como admite pocos bebés por turno, realizó un sorteo entre los postulantes. Para el turno de la mañana sorteó 8 titulares y para el turno de la tarde sorteó 5 titulares.
Escriba un script que realice el sorteo de 1 suplente para la mañana y 2 suplentes para la tarde. Diseñe una función que reciba los datos de estos niños y redacte un comunicado para avisar a los padres que resultaron sorteados, indicando el orden de sorteo y turno.

# Solución -> Funciones

```
import random
def redactar_correo(turno,**bebes):
  for t,d in bebes.items():
    print("Estimados papis, su bebe",t,d,"fue aceptado en el turno ",turno)
```

# Solución -> Script

```
dni=[60987789,60123321,68765345,65345543,68111123,65765567,66543444]

x=random.randrange(0,7)
M_suplente1=dni[x]
dni.remove(dni[x])
redactar_correo("Mañana",suplente1=M_suplente1)

x=random.randrange(0,6)
T_suplente1=dni[random.randrange(0,6)]
dni.remove(dni[x])
x=random.randrange(0,5)
T_suplente2=dni[random.randrange(0,5)]
dni.remove(dni[x])
redactar_correo("Tarde",suplente1=T_suplente1,suplente2=T_suplente2)
```

# Manejo de Excepciones

Llamamos excepciones a los errores generados por un código fuente. Si alguna función del programa genera un error y ésta no lo maneja, el error se propaga hasta llegar a la función principal que la invocó y genera que el programa se detenga.

Manejar los errores permite mostrar un error personalizado al usuario en vez de los clásicos errores del intérprete Python.

Para ello recurrimos a ciertas palabras reservadas que nos permiten realizar algunas acciones antes de detener nuestro programa por completo.

Los errores pueden provenir de un error de cálculos o del ingreso de un dato del usuario que nuestro código no es capaz de procesar, entre otros.

# Ejemplo:

Función o Script

```
a = 5
b = 0
print("a/b = ",a/b)
```

Salida generada

```
----> 3 print("a/b = ",a/b)
ZeroDivisionError: division by zero
```

# Ejemplo:

Función o Script

```
a = 5
b = 0
if b!=0:
    print(a/b)
else:
    print("No se puede dividir!")
```

Salida generada

```
No se puede dividir!
```

***Una primera aproximación al control de excepciones, podría ser realizar una comprobación manual de que no estamos dividiendo por cero, para así evitar el error.***

# Ejemplo:

Función o Script

```
try:
  a = 2
  b = 0
  print(a/b)
except:
  print("¡Error! ! ! ")
```

Salida generada

```
¡Error! ! !
```

Otra sintaxis para el manejo de excepciones es: **try / except**

**Ejemplo:** Otra sintaxis para el manejo de excepciones es: **try / except**

```
try:
  a = 2
  b = 0
  print(a/b)
except ZeroDivisionError:
  print("¡Error! ! ! ")
```

```
¡Error! ! !
```

Después de la palabra clave **except** debemos indicar el tipo de excepción que deseamos detectar.

A partir de la sentencia **try**, se tendrá en cuenta cualquier línea de código y si se produce una excepción serán ejecutadas las sentencias de código que aparecen dentro de la sentencia **except**.