Introducción a la programación con Python
Recursión y Manejo de errores

Alexis Rodríguez
Marcel Morán C

Esquema

- Pasar argumentos por funciones
- ¿Qué es recursión?
- ¿Qué es manejo de errores?
- Sintaxis de manejo de errores
- Tipo de Excepciones

Pasar argumentos por funciones

```
def calcular_peso(masa, gravedad, planeta):
    peso = gravedad*masa
    print("Tu peso en " + planeta + " con una masa de " + str(masa) + "es: ")
    print(peso, "N")

objecto = 20 # unidad kg
gravedad = 9.8 #unidad ms^2
planeta = "tierra"
peso_objecto = calcular_peso(objecto,gravedad,planeta)
```

Resultado:

Tu peso en tierra con una masa de 20 es:
196.0 N

Pasar argumentos por funciones

```
def calcular_peso(masa, gravedad, planeta):
    peso = gravedad*masa
    print("Tu peso en " + planeta + " con una masa de " + str(masa) + "es: ")
    print(peso, "N")

objecto = 20 # unidad kg
gravedad = 9.8 #unidad ms^2
planeta = "tierra"
peso_objecto = calcular_peso(planeta,gravedad,objecto)
```

Resultado:

TypeError: can't multiply sequence by non-int of type 'float'

Pasar argumentos por funciones

```
def calcular_peso(masa, gravedad, planeta):
    peso = gravedad*masa
    print("Tu peso en " + planeta + " con una masa de " + str(masa) + " es: ")
    print(peso, "N")

objecto = 20 # unidad kg
gravedad = 9.8 #unidad ms^2
planeta = "tierra"
peso_objecto = calcular_peso(planeta=planeta,gravedad=gravedad,masa=objecto)
```

Resultado:

Tu peso en tierra con una masa de 20 es:
196.0 N

Pasar argumentos por funciones

```
def calcular_peso(masa, planeta, gravedad=9.8):
    peso = gravedad*masa
    print("Tu peso en " + planeta + " con una masa de " + str(masa) + " es: ")
    print(peso, "N")

objecto = 20 # unidad kg
planeta = "tierra"
peso_objecto = calcular_peso(planeta=planeta,masa=objecto)
```

Resultado:

Tu peso en tierra con una masa de 20 es:
196.0 N

Pasar argumentos por funciones

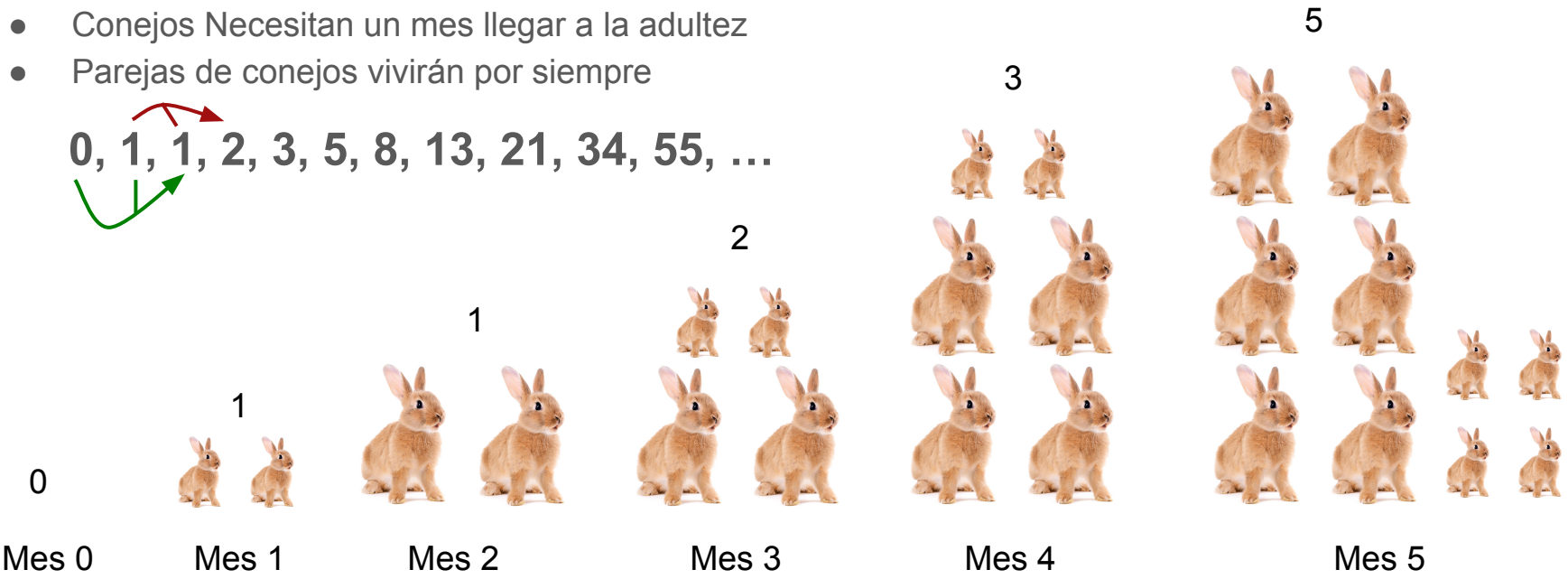
```python
def suma(*numeros):
    total = 0
    for num in numeros:
        total += num
    print(total)

suma(1,2,3)
```

Resultado: 6

La secuencia de Fibonacci

- Partiendo de una pareja recién nacida de conejos, cuántas parejas al final de un año podemos conseguir
- Solo pareja de conejos adultos pueden generar otra pareja
- Conejos Necesitan un mes llegar a la adultez
- Parejas de conejos vivirán por siempre

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …



La secuencia de Fibonacci

- 0, 1, 1, **2,** 3, 5, 8, 13, 21, 34, 55, …

```
fibo_seq_index = 3
seq_index = 0
num_en_seq = 0
ultimo_numero = 0
penultimo = 0

while(seq_index <= fibo_seq_index):
  if(seq_index <= 1):
      num_en_seq = seq_index
      ultimo_numero = seq_index
      penultimo = 0
  else:
      num_en_seq = ultimo_numero + penultimo
      penultimo = ultimo_numero
      ultimo_numero = num_en_seq
  seq_index += 1

print("El numero del termino " + str(fibo_seq_index) + " es " +  str(num_en_seq))
```

Resultado:
El numero del termino 3 es 2

¿Qué es recursión?

- Es una manera de resolver un **problema complejo**, dividiéndolo en **problemas más fáciles** usando una función que se llama a sí misma
- Por ejemplo, La secuencia de Fibonacci
- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
- Caso base
- Caso n+1
  f(0) = 0
  f(1) = 1

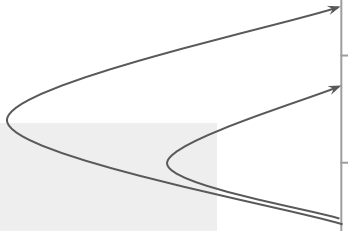
```
def fibonacci_seq(termino):
  if(termino <= 1):
    return termino
fibonacci_seq(0)
fibonacci_seq(1)
```

Entrada	Salida
0	0
1	1

¿Qué es recursión?

- Caso n+1
- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
  f(2) = f(1) + f(0)
  f(3) = f(2) + f(1)

```
def fibonacci_seq(termino):
  if(termino <= 1):
    return termino
  else:
    return fibonacci_seq(termino-1) + fibonacci_seq(termino-2)
fibonacci_seq(2)
fibonacci_seq(3)
```

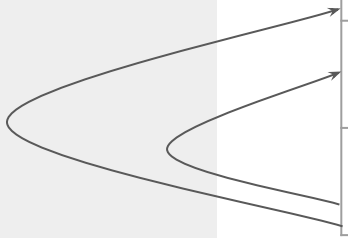


Entrada	Salida
0	0
1	1
2	1
3	

¿Qué es recursión?

- Caso n+1
- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
  f(2) = f(1) + f(0)
  f(3) = f(2) + f(1)

```
def fibonacci_seq(termino):
  if(termino <= 1):
    return termino
  else:
    return fibonacci_seq(termino-1) + fibonacci_seq(termino-2)
fibonacci_seq(2)
fibonacci_seq(3)
```

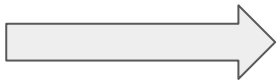


Entrada	Salida
0	0
1	1
2	1
3	**2**

¿Qué es manejo de errores?

```
def division ejemplo(numero1, numero2):
    resultado = numero1 / numero2
    print('Resultado de la division es:', resultado)
```

```
division_ejemplo(4, 2)
```



```
Resultado de la division es: 2.0
```

```
division_ejemplo(4, 0)
```

```
ZeroDivisionError                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_24784/1793463972.py in <module>
      3     print('Resultado de la division es:', resultado)
      4
----> 5 division_ejemplo(2, 0)
~\AppData\Local\Temp/ipykernel_24784/1793463972.py in division_ejemplo(numero1, numero2)
      1 def division_ejemplo(numero1, numero2):
----> 2     resultado = numero1 / numero2
      3     print('Resultado de la division es:', resultado)
      4
      5 division_ejemplo(4, 0)
ZeroDivisionError: division by zero
```

Sintaxis de manejo de errores

```
def division_ejemplo(numero1, numero2):
    try:
        resultado = numero1 / numero2
        print('Resultado es:', resultado)
    except ZeroDivisionError:
        print('Error: No es posible dividir para zero')
```

```
division_ejemplo(4, 2)
```

```
Resultado de la division es: 2.0
```

```
division_ejemplo(4, 0)
```

```
Error: No es posible dividir para zero
```

Tipos de excepciones

```
mistring = '45'
minumero = int(mistring)
print('mi numero es:', minumero)
```

```
mi numero es: 45
```

```
mi string = 'hello'
mi numero = int(mi string)
print('mi numero es:', mi_numero)
```

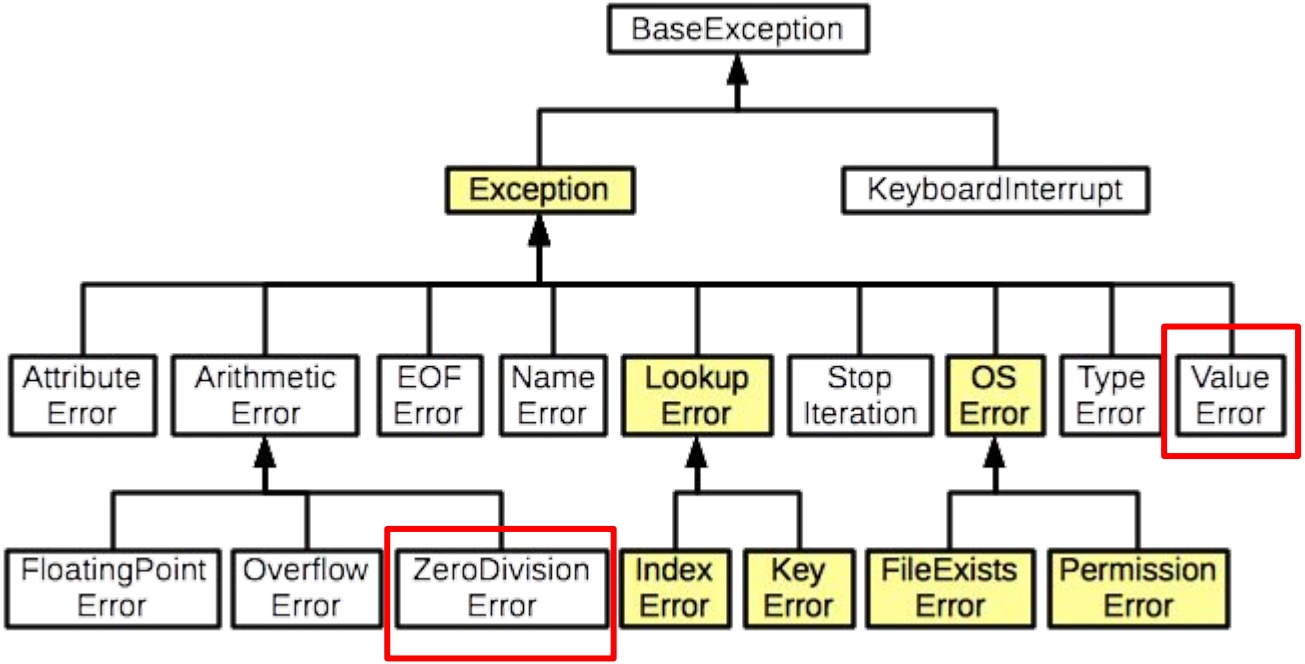
```
---------------------------------------------
-----------------------------

ValueError
Traceback (most recent call last)

~\AppData\Local\Temp/ipykernel_24784/2402285655
.py in <module>
      1 mi string = 'hello'
----> 2 mi_numero = int(mi_string)
      3 print('mi numero es:', mi_numero)

ValueError: invalid literal for int() with
base 10: 'hello'
```

Tipos de excepciones

```
mi_string = 'hello'
try:
    mi_numero = int(mi_string)
    print('mi numero es:', mi_numero)
except ValueError:
    print('Error: Valor invalido', "'" + mi_string + "'")
```

```
Error: Valor invalido 'hello'
```

Tipos de excepciones



Tipos de excepciones

```
try:
  ...
except PrimeraExcepcion:
  manejar_primera()

except SegundaExcepcion:
  manejar_segunda()

except (TerceraExcepcion, CuartaExcepcion, QuintaExcepcion)  as e:
  manejar_3ra_4ta_5ta()

except Exception:
  manejar_todo_lo_demas()
```

Incluye a todos los tipos de excepciones

Conclusión

- Otras maneras de pasar argumentos
- Recursión utiliza llamadas a una misma función para resolver problemas
- Cómo manejar errores
- Sintaxis de excepciones (try, except) y los tipos que existen

Retroalimentación

- Para retroalimentación dirigirse al siguiente enlace https://forms.gle/UEtzxHteT6NKYipj8 .
- Déjanos saber qué podemos hacer para mejorar el curso