



## Python Avanzado



## Herencia

- Nos permite reutilizar código de una clase en otra.
- Permite que un objeto sea de más de un tipo.
- En python3 todos los objetos heredan de la clase object.
- Se indica en la definición de la clase, de cuál hereda.



```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * self.length + 2 * self.width

class Square(Rectangle):
    def __init__(self, length):
        super().__init__(length, length)
```



```
class Square (Rectangle) :  
    def __init__(self, length):  
        super().__init__(length, length)  
  
    def area(self): #override de método  
        return 10 + super().area()
```



## **Herencia: Ejercicio**



## Captura de errores

- Errores de sintaxis:
  - Son errores del parser, debido a algo mal escrito (falta de dos puntos, mal tabulado etc.)
- Excepciones:
  - Es un error que se produce durante la ejecución del código.
  - Se puede capturar y determinar qué porción de código ejecutar en dicho caso.



## Excepciones

```
def funcion1 (arg) :  
    #...  
    a = int (arg)  
    #...  
    return a
```

```
def funcion2 (arg) :  
    return funcion1 (arg)
```

```
def funcion3 (arg) :  
    return funcion2 (arg)
```

```
#inicio programa  
funcion3 ("hola")
```



## Excepciones

```
Traceback (most recent call last):  
  File "pruebaTry.py", line 11, in <module>  
    funcion3("hola")  
  File "pruebaTry.py", line 8, in funcion3  
    return funcion2(arg)  
  File "pruebaTry.py", line 5, in funcion2  
    return funcion1(arg)  
  File "pruebaTry.py", line 2, in funcion1  
    return int(arg)  
ValueError: invalid literal for int() with base  
10: 'hola'
```





## Algunos tipos

<https://docs.python.org/3.5/library/exceptions.html>

```
>>> 10 * (1/0)
```

```
Traceback (most recent call last):  
File "<stdin>", line 1, in <module>  
ZeroDivisionError: division by zero
```

```
>>> '2' + 2
```

```
Traceback (most recent call last):  
File "<stdin>", line 1, in <module>  
TypeError: Can't convert 'int' object to str  
implicitly
```



## Captura de errores

**try:**

```
number = int(input("Ingresa un numero: "))  
print("Ingresaste:" + str(number))
```

**except ValueError as e:**

```
print("Numero incorrecto")  
print(e)
```



## Captura de errores

```
try:
    some_function()
except Exception as e:
    # handle errors
finally:
    # cleanup
```



## Captura de errores

```
>>> try:
...     print('hi')
... except Exception as e:
...     print('Error')
... else:
...     print('Success')
... finally:
...     print('at last')
```

```
hi
Success
at last
```



## Lanzar excepciones

```
def imprimir_temp(self) :  
    if self.temp>100:  
        raise Exception("Temp fuera de rango")  
    else:  
        print(self.temp)
```



## Ejemplo Signals



```
import signal
```

```
signal.signal(signal.SIGINT, handler)
```

- Configurar handlers en Thread ppal.
- El handler se ejecutará en el thread ppal siempre.



```
def handler(sig, frame):    # define the handler
    print(sig)
    traceback.print_stack(frame)
```

- El campo frame nos permite hacer debug de dónde fue interrumpido el programa.





## Ejemplos Threads



- Importamos threading.
- Heredamos de la clase Thread.
- Reescribimos el método run().
- El método run() se ejecutará en otro thread.
- Para lanzarlo, ejecutamos el método start().
- Esperamos su finalización con join()



- Objeto **Event**

- Permite utilizarlo como flag para terminar threads.

```
self.shutdown_flag = threading.Event()
```

```
shutdown_flag.set() # levanta flag
```

```
shutdown_flag.is_set() #consulta flag
```



## Ejemplos Sockets



## JSON encode y decode

```
import json
```

```
json.dumps(dicc o lista) #devuelve un string
```

```
json.loads(texto json) #devuelve dicc o lista
```



## Bibliografía

Gowrishankar S. Veena A. (2019). Introduction to Python Programming. NW. Taylor & Francis Group, LLC.

Matt Harrison. Illustrated guide to python 3. (2017).  
Treading on Python Series

<https://pip.pypa.io/en/stable/installing/>

<https://www.learnpython.org/es/>

<https://docs.python.org/3/library/stdtypes.html>