Errores y excepciones

- Error de sintaxis

```
def func() # <- faltan los dos puntos
    print("hi")
```

- Error por variables indefinidas

```
print(var) # -> no ha sido definida
```

- Error de división por cero

```
# Por ejemplo al dividir 1/0
```

- Por tipo de dato

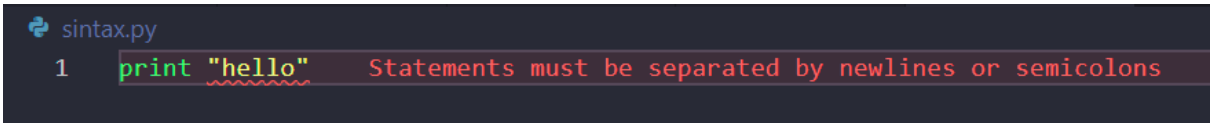
```
one = 1
onetxt = "one"

res = one + onetxt # Dará error, int y string no pueden sumar
```

Errores de sintaxis

- Los resalta el editor

- Los muestra la consola tras la ejecución
- En consola aparece como *"SyntaxError"*




```
File "C:\...\python\test\sintax.py", line 1
    print "hello"
    ^^^^^^^^^^^^^
SyntaxError: Missing parentheses in call to 'print'. Did you
```

Excepciones

Ocurren en tiempo de ejecución del programa, aunque el código esté escrito correctamente.

Raise exception

Ejemplo de función para evaluar un número:

```
def validar_x(x):
    if x < 1:
        raise Exception("La variable x debe ser mayor que 1
    else:
        print(f"{x} es mayor que 1")
```

Si `x` vale `-1`:

```
Traceback (most recent call last):
  File "C:\...\python\test\sintax.py", line 7, in <module>
    validar_x(-1)
    ~~~~~~~~~^^^^
```

```
  File "C:\Users\Desire\Desktop\APUNTES\python\test\sintax.
    raise Exception("La variable x debe ser mayor que 1")
Exception: La variable x debe ser mayor que 1
```

Aparecerá la excepción en la consola.

Assert

Añadimos un assert en una función para calcular una media de notas que recibe una lista, para comprobar si la lista tiene valores o no. Si está vacía, lanzará un assert.

```
def calc_avg(list):
    """ Función que recibe una lista y calcula el promedio
    assert len(list) > 0, "La lista está vacía" # Comprobar
    return sum(list) / len(list)

avg = calc_avg(list=[]);
```

Si la lista que enviamos está vacía:

```
Traceback (most recent call last):
  File "C:\...\python\test\sintax.py", line 5, in <module>
    avg = calc_avg(list=[]);
  File "C:\Users\Desire\Desktop\APUNTES\python\test\sintax.
    assert len(list) > 0, "La lista está vacía"
           ^^^^^^^^^^^^^
AssertionError: La lista está vacía
```

Try - Except

- Para capturar errores y manejarlos según nuestras necesidades
- En el bloque `try` se encuentra el código que debe ejecutarse en el programa

- En el bloque `except` están las instrucciones a ejecutar si algo en el bloque `try` falla

```
def calc_avg(list):
    """ Función que recibe una lista y calcula el promedio
    return sum(list) / len(list)

try:
    avg = calc_avg([])
    print(avg)
except:
    print("La ejecución ha fallado")
```

La ejecución finalizará sin errores en consola, pero lanzará la excepción.

En cambio, si la lista sí tiene valores, la ejecución se realizará sin problema:

```
def calc_avg(list):
    """ Función que recibe una lista y calcula el promedio
    return sum(list) / len(list)

try:
    avg = calc_avg([1, 2, 3, 4])
    print(avg) # Imprimirá 2.5
except:
    print("La ejecución ha fallado")
```

El bloque `except` nos permite capturar un error concreto.

```
def calc_avg(list):
    """ Función que recibe una lista y calcula el promedio
    return sum(list) / len(list)

try:
    avg = calc_avg([])
    print(avg)
except Exception as e:
```

```
    print("La ejecución ha fallado")
    print(e) # Imprimirá: La ejecución ha fallado - divisic
```

Si añadimos de nuevo el assert en la función, podemos capturarlo dentro de un `except` de tipo `AssertionError`:

```
def calc_avg(list):
    """ Función que recibe una lista y calcula el promedio
    assert len(list) > 0, "La lista está vacía" # Comprobar
    return sum(list) / len(list)

try:
    avg = calc_avg([])
    print(avg)
except AssertionError as ae:
    print(ae) # Imprimirá: La lista está vacía
except Exception as e:
    print("La ejecución ha fallado")
    print(e)
```

El bloque `except AssertionError as ae:` sólo capturará un error de `assertion` cuando este suceda dentro de la función `calc_avg`.

Cualquier otro tipo de error desconocido o no definido en nuestro bloque `except`, caerá en el bloque definido como `except Exception`:

```
def calc_avg(list):
    """ Función que recibe una lista y calcula el promedio
    assert len(list) > 0, "La lista está vacía" # Comprobar
    return sum(list) / len(list)

try:
    avg = calc_avg(["texto"])
    print(avg)
except AssertionError as ae:
    print(ae) # Imprimirá: La lista está vacía
except Exception as e:
```

```
    print("La ejecución ha fallado")
    print(e)
```

@Desiré Marrón