

# Taller de Programación

## Clase 10: Manejo de Errores

Daniela Opitz, Leo Ferres  
[dopitz@udd.cl](mailto:dopitz@udd.cl)

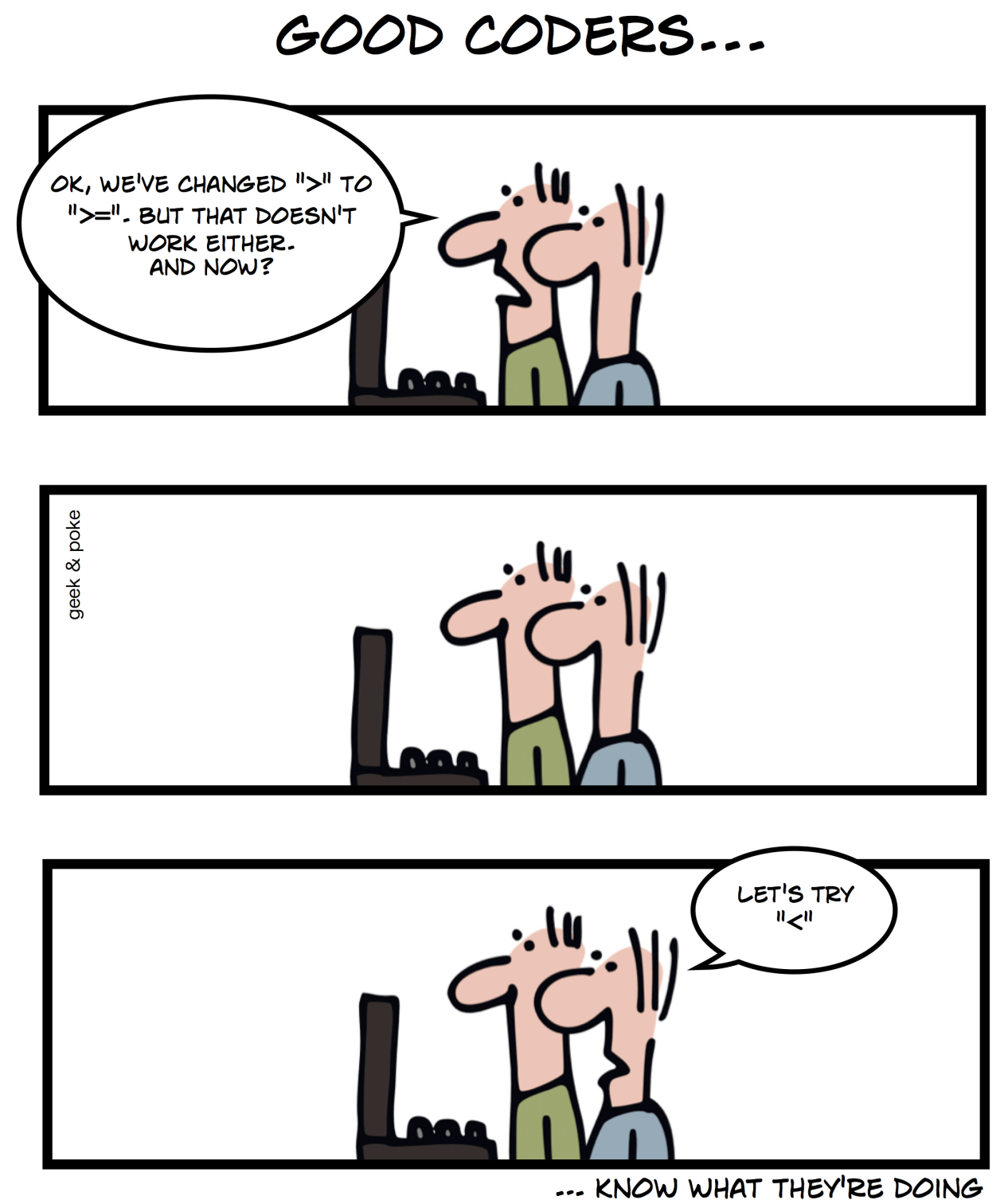


Basada en presentaciones oficiales de libro Introduction to Programming in Python (Sedgewick, Wayne, Dondero).

Disponible en <https://introcs.cs.princeton.edu/python>

# Errores y Excepciones

- **Syntax errors:** el código no es válido (fácil de arreglar)
- **Runtime errors:** código es sintácticamente válido, pero no ejecuta ()
- **Semantic errors:** código sintácticamente válido y que se ejecuta pero que no hace lo que el programador quería que hiciera.



# Runtime

```
print(Q)
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-1-e796bdcf24ff> in <module>()  
----> 1 print(Q)
```

NameError: name 'Q' is not defined

```
1 + 'abc'
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-2-aab9e8ede4f7> in <module>()  
----> 1 1 + 'abc'
```

TypeError: unsupported operand type(s) for +: 'int' and 'str'

```
2 / 0
```

```
-----  
ZeroDivisionError                        Traceback (most recent call last)  
<ipython-input-3-ae0c5d243292> in <module>()  
----> 1 2 / 0
```

ZeroDivisionError: division by zero

```
L = [1, 2, 3]  
L[1000]
```

```
-----  
IndexError                                Traceback (most recent call last)  
<ipython-input-4-06b6eb1b8957> in <module>()  
      1 L = [1, 2, 3]  
----> 2 L[1000]
```

IndexError: list index out of range

# Try and except

- Ejemplo 1

```
try:  
    print("Esto se ejecuta primero")  
except:  
    print("Esto se ejecuta solo si hay un error")
```

El segundo bloque de código no se ejecuta porque el primero no reportó ningún error.

- Ejemplo 2

```
try:  
    print("Intentemos dividir por cero")  
    x = 1 / 0 # ZeroDivisionError  
except:  
    print("Algo malo ocurrió")
```

# Try and except

¿Qué pasa al ejecutar?

```
def safe_divide(a, b):  
    try:  
        return a / b  
    except:  
        return 1E100
```



```
safe_divide(1, 2)
```

```
safe_divide(2, 0)
```

# Raising Exceptions

Obviamente, es bueno tener excepciones que son informativas. Hay veces que también queremos agregar (**raise**) nuestras propias excepciones!

```
raise RuntimeError("my error message")
```

```
-----  
RuntimeError                                Traceback (most recent call last)  
<ipython-input-16-c6a4c1ed2f34> in <module>()  
----> 1 raise RuntimeError("my error message")  
  
RuntimeError: my error message
```

**RunTimeError**

# Raising Exceptions

```
def fibonacci(N):  
    L = []  
    a, b = 0, 1  
    while len(L) < N:  
        a, b = b, a + b  
        L.append(a)  
    return L
```

```
def fibonacci(N):  
    if N < 0:  
        raise ValueError("N must be non-negative")  
    L = []  
    a, b = 0, 1  
    while len(L) < N:  
        a, b = b, a + b  
        L.append(a)  
    return L
```



# Raising Exceptions

`fibonacci(-10)`



```
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-20-3d291499cfa7> in <module>()  
----> 1 fibonacci(-10)  
  
<ipython-input-18-01d0cf168d63> in fibonacci(N)  
      1 def fibonacci(N):  
      2     if N < 0:  
----> 3         raise ValueError("N must be non-negative")  
      4     L = []  
      5     a, b = 0, 1
```

ValueError: N must be non-negative

Ahora el usuario sabe que el "input" es inválido. Y hasta podemos usar un `try...except`