In [ ]:
```
#  Esta celda es exclusivo para cuestiones de impresion dentro del notebook
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

Flujo de condiciones y bucles

Que aprenderemos?

- condicionales
- clase `bool`
- bucles `while` y `for`
- tips para iterar de manera 'pythonica'

Condicionales

`if`, `elif` y `else`

In [ ]:
```
#  ejemplo de condicionales
resultado = 0.5
if resultado.is_integer(): # la condicion es verdadera
    print('El resultado es entero')
else: # la condicion anterior es falso
    print('El resultado es flotante')
```

In [ ]:
```
#  ejemplo con elif
linea = 'Introduccion A Python'
if linea.istitle():
    print(f'{linea!r} es un titulo')
elif linea.isupper():
    print(f'{linea!r} esta en mayusculas')
else:
    print(f'{linea!r} esta en minusculas')
```

Clase `bool`

Representacion literal

In [ ]:
```
condicion = True
otra_condicion = False
```

In [ ]:
```
#  bool es una subclase de los enteros
True == 1
False == 0

issubclass(bool, int)
```

In [ ]:
```
lista = []
#  verificar que una lista esta vacia
if len(lista) == 0:
    print('La lista esta vacia')

#  modo pythonico
if not lista:
    print('La lista esta vacia')
```

El valor 0, y secuencias vacias('', [], {}, set(), etc) son valores falsos, cualquier numero diferente de 0 y secuencias no vacias son verdaderas, por lo que no debemos hacer comparaciones con booleanos

Operador ternario

In [ ]:
```
suma = 10 + 7
respuesta = 'Impar' if suma%2 else 'Par'
```

switch - case ?

In [ ]:
```
#  simulando un switch case con un dict
from operator import add as suma, sub as resta,\
    mul as multiplicacion, truediv as division
opcion = 'suma'
decision = {'suma': suma,
            'resta': resta,
            'multiplicacion': multiplicacion,
            'division': division}.get(opcion)
decision(4,2) # decision guarda una funcion, que puede ser ejecutado posteriormente
```

funciones `any` y `all`

In [ ]:
```
condiciones = [True, True, False]
any(condiciones) #  Devuelve True si un elemento es True
all(condiciones) #  Devuelve True si todos los elementos son True
```

Bucles

`while`

In [ ]:
```
contador = 0
#  El bucle se ejecuta hasta que la condicion sea falsa
while contador < 10:
    print(contador)
    contador += 1
```

`for`

In [ ]:
```
for i in range(10):
    print(i)
```

In [ ]:
```
pares = [2,4,6,8,10]
#  iteracion por indices
for i in range(5):
    print(pares[i])

#  recorriendo un iterable
for par in pares:
    print(par)
```

In [ ]:
```
#  imprimir indice y el elemento en el mismo
for i in range(5):
    print(f'{i} {pares[i]}')

#  lo mismo, pero mas pythonico XD
for i, par in enumerate(pares):
    print(f'{i} {par}')
```

class `range`

In [ ]:
```
a = range(10) #  stop
list(a)
b = range(0,10) #  start, stop
list(b)
c = range(0,10,2) #  start, stop, step
list(c)
```

Interrupir una iteracion (`continue` y `break`)

In [ ]:
```
#  continue omite todo el codigo posterior a el, y continua con la siguiente iteracion
for i in range(30):
    if not i%3:
        continue
    print(i)
```

In [ ]:
```
#  break termina con el bucle
for j in range(10):
    print(j)
    if j == 5:
        break
```

Clausula `else` de los bucles

In [ ]:
```
for j in range(11):
    print(j)
    if j == 5:
        break
else: #  ademas
    #  el bloque de codigo se ejecuta cuando el bucle
    #  no es interrumpido
    print('No fue encontrado un 5')
```

`iter` y `next`

In [ ]:
```
iterador = iter('Python 3.7') #  objeto
for letra in iterador:
    print(letra, sep='')
print()

#  solo se puede iterar una vez!
for letra in iterador:
    print(letra, sep='')
print()
```

In [ ]:
```
archivo = open('ejemplo.txt')
it_archivo = iter(archivo.readline, '/') #  invocable, sentinela
next(it_archivo)
next(it_archivo)
next(it_archivo)
next(it_archivo)
```

Menciones

- `zip`