Loops

Loops versus conditional expressions

- If…elif…else allows us to execute code **conditionnaly**
- The code within an if structure is executed **at most once**
- We need a way to **repeat** the execution of pieces of code

The for loop

- for allows us to execute code a **predetermined** number of times

```
for n in range(4):
    print("----")
    print(n)
```

```
----
0
----
1
----
2
----
3
```

for loop

- for **variable** in **range(…):**
- variable changes value every time we go back to the top of the for loop
- range allows us to defined from which value to which value the code must be executed

Range

- range(a, b) goes from a **included** to b **excluded**
- range(b) is the same as range(0, b), from 0 to b – 1
- range(a, b, c) goes from a to b (excluded) by step of c

```
for n in range(0, 10, 3):
    print(n)
```

```
0
3
6
9
```

Example

```
print("T_f,    T_c")
for Tf in range(-100, 200, 20):
    print(Tf, (Tf - 32)*5/9)
```

```
T_f,    T_c
-100 -73.33333333333333
-80 -62.22222222222222
-60 -51.111111111111114
-40 -40.0
-20 -28.88888888888889
0 -17.77777777777778
20 -6.666666666666667
40 4.444444444444445
60 15.555555555555555
80 26.666666666666668
100 37.77777777777778
120 48.888888888888886
140 60.0
160 71.11111111111111
180 82.22222222222223
```

While loops

- for loops are great when we already know how many times we want to execute some code
- When we don't know in advance, we use a while loop
- while (boolean expression):

Example

```
print("Start while")
x = -2
while x < 5:
    print(x)
    x += 1  # Increment x
print("End while")
```

```
Start while
-2
-1
0
1
2
3
4
End while
```

Notes

- The example we just saw would work with a for loop
- This one only works with while

```
x = 0.9
while x > 0.001:
    # Computing x squared
    x = x*x
    print(x)
```

```
0.81
0.6561000000000001
0.43046721000000016
0.18530201888518424
0.03433683820292518
0.0011790184577738 62
1.390084523771456e-06
```

break, continue and pass

01

break allows us to exit a loop before its natural end

02

continue allows us to go back to the beginning of a loop without executing the code below the continue

03

pass is a placeholder for future code

break

```
for x in range(10):
    print(x)
    if x == 5:
        print("Time we exit this loop")
        break
```

```
0
1
2
3
4
5
Time we exit this loop
```

continue

```python
for j in range(20):
    if j % 4 == 0:  # checks if j is a multiple of 4
        continue  # skips to the next iteration of the loop
    print("This number cannot be divided by 4:", j)
```

```
This number cannot be divided by 4: 1
This number cannot be divided by 4: 2
This number cannot be divided by 4: 3
This number cannot be divided by 4: 5
This number cannot be divided by 4: 6
This number cannot be divided by 4: 7
This number cannot be divided by 4: 9
This number cannot be divided by 4: 10
This number cannot be divided by 4: 11
This number cannot be divided by 4: 13
This number cannot be divided by 4: 14
This number cannot be divided by 4: 15
This number cannot be divided by 4: 17
This number cannot be divided by 4: 18
This number cannot be divided by 4: 19
```

pass

```
for x in range(10):
    if x < 5:
        # TODO: implémenter le cas où x < 5 quand les autres cas sont traités
        pass
    elif x < 9:
        print(x*x)
    else:
        print(x)
```

```
25
36
49
64
9
```

Now that we can loop, we may want to do more with print

print can take two optional arguments : sep and end

sep stands for separator, its default value is a space

end is empty by default – it is what is displayed at the end

Going further with print

Example

```
1 print("foo", "bar", sep="-", end="!")
```

```
foo-bar!
```

Conclusion

Loops are **fundamental** to write powerful code

Loops can be used inside if statements, if statements can be used inside loops

We can even have **loops inside loops**!

Practice using loops and conditional structures, use your imagination and **unleash all their potential!**