Loops

Loops

- Loops are used to **repeat** a block of **code** multiple times.
- They allow you to automate repetitive tasks and process large amounts of data.
- There are two main types of loops in Python:

while loops	for loops
Repeat code based on: Some **condition**	Repeat code based on: **Elements** in an enumerable (e.g. a list)

For loops

`for` loops are used to iterate over a **sequence of elements**.

Syntax:

```
for item in sequence:

    # Code to execute for each item in the sequence
```

For loops

Example:

```
numbers = [1, 2, 3, 4, 5]

for number in numbers:

        print(number)
```

Try it out yourself! Does it work?

What happens if you use a **set** instead?

For loops

Type the following code snippet. What does it output?

```
for number in range(10):
        print(number)
```

What does `range()` do?

Play around with the code. Replace **`range(10)`** with the following. What does it do?

- `range(5, 10)`
- `range(1, 10, 2)`

start *end* *steps*

Exercise

1. Create an **empty list** named **`numbers`**
2. Using for loop, write a Python program that prompts the user to enter the **length of the list** and then **fills up the list `numbers` with numbers from 1 to the length** entered by the user.

```
length = int(input("Enter the length of the list: "))

numbers = …

# Add your for loop here

print(numbers)
```

```
Enter the length of the list: 5
[1, 2, 3, 4, 5]
```

Exercise Solution *(just one possibility!)*

```
length = int(input("Enter the length of the list: "))
numbers = []

# Fill up the list with numbers using a for loop
for i in range(1, length + 1):
    numbers.append(i)

print(numbers)
```

```
Enter the length of the list: 5
[1, 2, 3, 4, 5]
```

While Loops

While loops are used to **repeat** a block of code as long as a **condition** is True.

Syntax:

```
while condition:
        # Code to execute as long as the condition is true
```

While Loops

While loops are used to **repeat** a block of code as long as a **condition** is True.

Syntax:

must evaluate to True or False

```
while condition:
        # Code to execute as long as the condition is true
```

While Loops

Example:

```
x = int(input())

while x <= 5:

        print(x)

        x += 1
```

Question: What is the output of the above Program?

Exercise

Write a Python program that **counts down from 10 to 1** and then **prints "Time's up!"**.

```
seconds = 10

# Insert your while loop here!

print("Time's up!")
```

```
10
9
8
7
6
5
4
3
2
1
Time's up!
```

Exercise Solution

```
seconds = 10
while seconds > 0:
    print(seconds)
    seconds -= 1
print("Time's up!")
```

BONUS

Use the internet to find out how to make python **wait** for a second (`sleep`).
Modify the solution to **actually count** down!

Break and continue

`break` - cancels the loop

`continue` - skips to the next iteration

Example:

```
for i in range(10):
        if i == 2:
                continue
        print(i)
```

What does this program output?

Break and continue

`break` - cancels the loop

`continue` - skips to the next iteration

Example:

```
for i in range(10):
        if i == 2:
                continue
        print(i)
```

```
0
1
3
4
5
6
7
8
9
>>>
```

Break and continue

`break` - cancels the loop

`continue` - skips to the next iteration

Example:

```
for i in range(10):
        if i == 2:
                break
        print(i)
```

What does this program output?

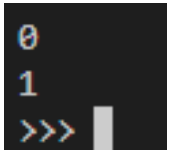
Break and continue

`break` - cancels the loop

`continue` - skips to the next iteration

Example:

```
for i in range(10):
        if i == 2:
                break
        print(i)
```



Break and continue

`break` - cancels the loop

`continue` - skips to the next iteration

Example:

```
for i in range(10):
        if i == 2:
                break
        print(i)
```

NOTE

`break` and `continue` also works with `while`!

Recap

If statement:

```
x= int(input())
if x > 0:
    print("positive")
elif x == 0:
    print("Negative")
else:
    print("Zero")
```

while loop:

```
x = 0
while x <= 10 :
        print(x)
        x += 1
```

for loop:

```
for i in range(10+1) :
        print(x)
```

`break` - cancels the loop

`continue` - skips to the next iteration