9) Conditions and loops

Agenda

- While loop
- For loop
- Bigger/smaller than
- Import

Guess the secret number

Homework from the previous lesson was to build a game called **Guess a secret number**. If you did it correctly, it looks something like this:

```
secret = 22

guess = int(input("Guess the secret number (between 1 and 30): "))

if guess == secret:
    print("You've guessed it - congratulations! It's number 22.")
else:
    print("Sorry, your guess is not correct... The secret number is not " + str(guess))
```

As you see in the last line, it's possible also to **convert a number into a string** (using the `str()` function).

Make user guess more than once

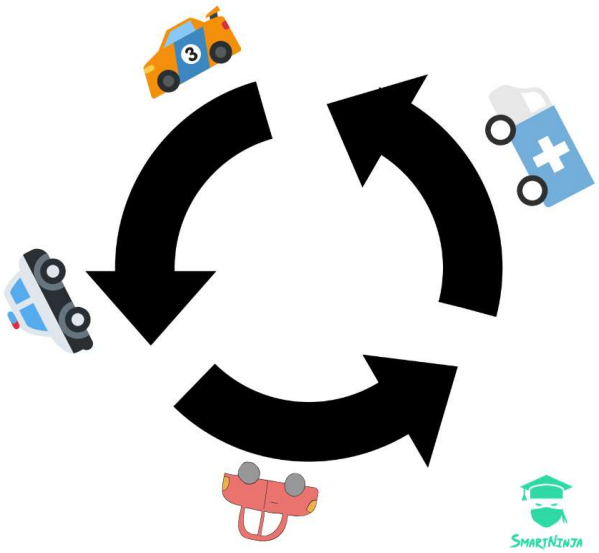
So far our game let's you **guess only once**. We'd like to give our user ability to guess multiple times. To achieve this, we have to use **loops**.

When you use a loop it means that some part of the code (that's in the loop) will **constantly run**, **until some condition is met**.

You can imagine a loop as a roundabout.

You drive into a roundabout in some foreign city and you're trying to figure out where you have to go next.

Until you figure that out (condition), you are circling in the roundabout. When you finally figure it out (or your GPS starts working), you go out of the loop.



There are two types of loops: **while loop** and **for loop.**

While loop

Let's first use while loop in our program. The **basic logic** is this: keep asking the user what the secret number is (loop) until user makes the right guess (condition).

```
secret = 22
guess = 0

while guess != secret:
    guess = int(input("Guess the secret number (between 1 and 30): "))

    if guess == secret:
        print("You guessed it - congratulations! It's number 22.")
    else:
        print("Sorry, your guess is not correct... The secret number is not " + str(guess))
```

It's a pretty simple implementation: while user's guess **is not equal** (`!=`) to the secret, **keep running** the code in the loop. When the guess **is equal** to secret, the code in the loop **doesn't run anymore**.

Note that we had to define `guess` as `0` first, otherwise we wouldn't be able to write `while guess != secret` . You **cannot use a variable before you declare it**.

In this example we keep asking the user what the secret number is, until the user makes the right guess. But what about if we want to **limit user's guess to 5 attempts**? The best way would be to use a **for loop** instead.

For loop

The difference between while and for loop is that with while loop **we cannot know how long it will run** - this means how many times the code will be repeated. It can run forever (so called infinite loop or endless loop), but this is something we usually don't want to do.

For loop, on the contrary, has **limited "repeats"**. In our case, for loop will run only 5 times. Let's implement this:

```
secret = 22

for x in range(5):
    guess = int(input("Guess the secret number (between 1 and 30): "))

    if guess == secret:
        print("You've guessed it - congratulations! It's number 22.")
        break
    else:
        print("Sorry, your guess is not correct... The secret number is not " + str(guess))
```

We added two things here:

- `for x in range(5)` : this means the code will run 5 times and each time the `x` variable will hold a different number (you can add `print x` to your code to see it).
- `break` : this is a special keyword that helps you **stop the loop**. In our case we want to stop the loop if the user guesses the right number (even if the loop hasn't had all the 5 cycles yet).

While + Break

This `break` thing is interesting, right? We could also use it in the while loop. In this case, we could write our program a bit differently:

```
secret = 22

while True:
    guess = int(input("Guess the secret number (between 1 and 30): "))

    if guess == secret:
        print("You've guessed it - congratulations! It's number 22.")
        break
    else:
        print("Sorry, your guess is not correct... The secret number is not " + str(guess))
```

We wrote our while loop a bit differently here. We used `while True` , which basically create that **endless loop** we mentioned before. This means our while loop will **run forever**. The only way to stop it is using the `break` keyword.

Hint: is your guess too big or too small?

Another thing to improve our little game would be to give our users a **hint if their guess was too big or too small**. Here we use simple **bigger than** (>) and **smaller than** (<) symbols:

```
secret = 22

while True:
    guess = int(input("Guess the secret number (between 1 and 30): "))

    if guess == secret:
        print("You've guessed it - congratulations! It's number 22.")
        break
    elif guess > secret:
        print("Your guess is not correct... try something smaller")
    elif guess < secret:
        print("Your guess is not correct... try something bigger")
```

We just added two new `elif` statements which **check** if the guess was bigger or smaller than the secret number and give user a **friendly hint**.

Random

The last thing to improve in our program is to have our computer to choose a random number as a secret. So far we had the number 22 written, but it will be more fun if our computer chooses a number on random.

To enable this, we'll have to import a library called `random`. Write this at the top of your program:

```
import random
```

What is a library? It's just a file with some code that some other developers wrote and which we can use in our projects. This way we don't have to reinvent the wheel.

Next change the value of the `secret` variable into this:

```
secret = random.randint(1, 30)
```

This means the computer will randomly select a number between 1 and 30 (including 1 and 30).

And because computer will select a random number each time, we have to change our "success" message into this:

```
print("You've guessed it - congratulations! It's number " + str(secret))
```

The whole program now looks like this:

```
import random


secret = random.randint(1, 30)

while True:
    guess = int(input("Guess the secret number (between 1 and 30): "))

    if guess == secret:
        print("You've guessed it - congratulations! It's number " + str(secret))
        break
    elif guess > secret:
        print("Your guess is not correct... try something smaller")
    elif guess < secret:
        print("Your guess is not correct... try something bigger")
```

Run the program and have fun playing your first Python game! ;)

Q&A

Any question?

If there's enough time after Q&A, students can start working on the following exercise and homework.

Exercise 9.1: Unit converter

> *Students can do this exercise in pairs.*

Create a program that **converts units**. More specifically, **kilometers into miles**.

First create a plan for this program. **Together with the instructor** define the steps the user will take when using the program.

> *Plan:*
>
> - *The program greets user and describes what's the purpose of the program.*
> - *The program asks user to enter number of kilometers.*
> - *User enters the amount of kilometers.*
> - *The program converts these kilometers into miles and prints them.*
> - *The program asks user if s/he'd want to do another conversion.*
> - *If yes, repeat the above process (except the greeting).*
> - *If not, the program says goodbye and stops.*

The program should constantly run as long as the user would want to do conversions.

When you finish, push your code on GitHub (https://github.com/) and share it on the SmartNinja forum.

> *Solution (https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-09/unit-converter)*

Homework 9.2: FizzBuzz

- User enters a number between 1 and 100
- FizzBuzz program starts to count to that number (it prints them in the Terminal). In case the number is divisible with 3, it prints "fizz" instead of the number. If the number is divisible with 5, it prints "buzz". If it's divisible with both 3 and 5, it prints "fizzbuzz".

Example:

```
Select a number between 1 and 100
16

1
2
fizz
4
buzz
fizz
7
8
fizz
buzz
11
fizz
13
14
fizzbuzz
16
```

A tip for this project:

How to find a division remainder

If a division remainder is 0, that means some number is divisible with another (without remainder).

Example:

```
print(15 % 5)
print(15 % 4)
```

15 is divisible with 5, so the remainder is 0. But 15 is not divisible with 4, so the remainder is not 0.

When you finish, push your code on GitHub (https://github.com/) and share it on the SmartNinja forum.

> *Solution (https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-09/fizzbuzz)*

Homework 9.3: Make string lowercase

Sometimes you'd like to make some string lowercase. For example, you have a string like this:

```
"Today Is A BeautiFul DAY"
```

And you'd like to make it like this:

```
"today is a beautiful day"
```

There is a very nice solution in Python to do this. Use Google search and find out how to do it.

> *Where would this come handy? For example if you ask user "Would you like to continue (yes/no)?", the user might respond: "yes", "Yes", "YES" or even "YeS". In this case, changing your user's response into lowercase letters would be very helpful in your if-else statement.*

When you finish, push your code on GitHub (https://github.com/) and share it on the SmartNinja forum.

Bonus: joining strings

There are many ways how to join two strings together or how to add data in a string.

Consider these two strings:

```
str_one = "Happy"
str_two = "Day"
```

We can join them with a plus:

```
print(str_one + str_two)  # result: HappyDay
```

We can add a string in-between:

```
print(str_one + " " + str_two)  # result: Happy Day
```

We can use `%s` to add them in a string:

```
print("%s %s" % (str_one, str_two))
```

Or, we can use the `format()` method, with placeholders like `{0}`:

```
print("{0} {1}".format(str_one, str_two))
```

The preferred approach is the last one, the `format()` method.

> *In programming, counting starts with 0. That's why `{0}` means the first string.*

Useful links

- Python while loop examples (https://www.w3schools.com/python/python_while_loops.asp)
- Python for loop examples (https://www.w3schools.com/python/python_for_loops.asp)