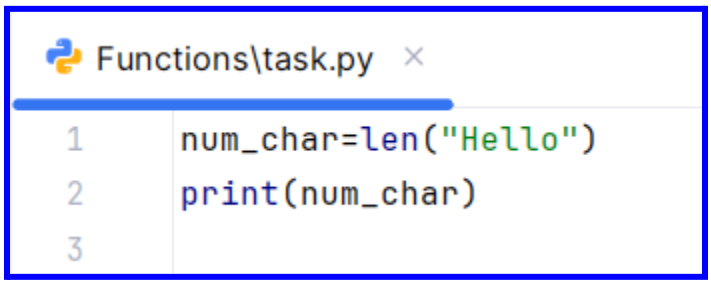
DAY 6: Python Functions & Karel

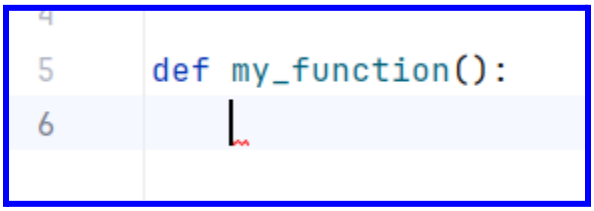
20/01/2025
Were going to see Code blocks and While loops and to learn all of these things were going to use and create something like Karel the robot: https://stanford.edu/~cpiech/karel/ide.html

Defining and calling Python functions

If you take a look of this document: https://docs.python.org/3/library/functions.html
then you can see that Python has a whole bunch of built in functions that we've been using.For example, the len() function, which gives us the number of items in a collection, or the int() function, and many ones that we didn't see yet.
We can type print () and the reason why we know it's a function is because it's the name of a function followed by a set of parentheses. Let's use the len function:



As you saw we save the function into a variable called num_char so then we can print it and see it in the output. So all of this functionality has been achieved by these built-in functions from Python like print() and len(). But what happens if we want to create our own function? How do we do that?
We start with a keyword called def after this we can put the name of our function:



the finishing touch to our function definition is a colon because that says everything that comes after that line and is indented belongs with the function.

```
5    def my_function():  1 usage
6        print("hello")
7        print("Bye")
8    my_function()
```

all we have to do is type the name of the function, which is my_function, and then to add the parentheses and any necessary inputs. This is to call it because if we don't do that it would not work our functions, it would not print anything. Here are the steps:

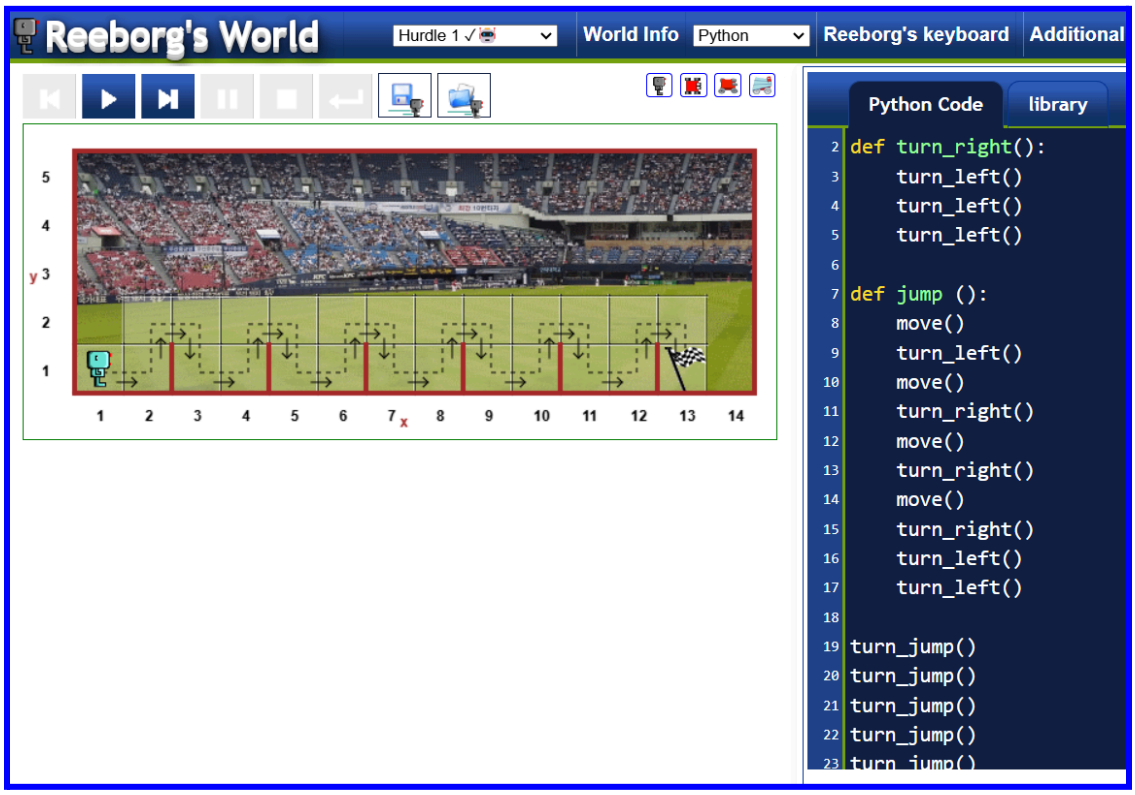
task.py

```
1
2    def my_function(): #we define our function and we give a name  1 usage
3        print("hello") #Then comes a set of parentheses and a colon
4        print("Bye")  #Thn we put the line of codes
5    my_function()     #Then we called the function
6
```

If we have a robot and we want to give instructions to him to everyday and pick up some milk we can't everyday type every instruction that's why we create variables in order to do allahabad of this steps close in one box and we just called the function and thats it, try this robot: https://reeborg.ca/reeborg.html?lang=en&mode=python&menu=worlds%2Fmenus%2Freeborg_intro_en.json&name=Alone&url=worlds%2Ftutorial_en%2Falone.json

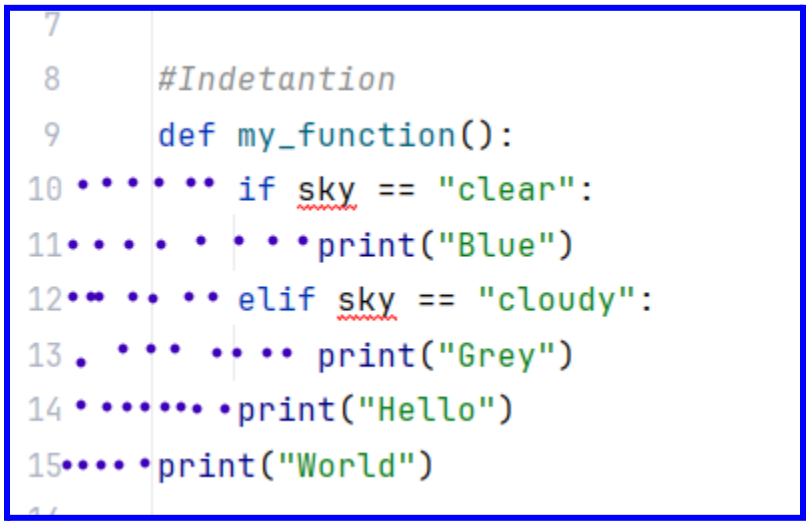
The Hurdles Loop Challenge

The idea is that we've got a robot here which needs to do a number of hurdles to jump over each of these barriers, to get to the final goal here.Try and see if you can minimize number lines of code while still keeping your code readable and understandable by somebody else and be able to get your robot



Indentation in Python

We've already mentioned how important indentation is in Python, so we know that when we create a function like this (def my_function), every line that comes after this definition that is indented is going to be inside this function. So by indented I mean it's shifted to the right by four spaces, if your variable is not indented the computer will read it separately, is like a file structure, think it like a folder and what's inside is the functions. Now the indentation gets a little bit more complicated when we have other blocks of code. So for example, the if, elif, else statements, they have blocks of code which need to be indented to be inside the block, for loops need to be indented to be inside the loop. And it's very important that you get used to looking at blocks of code like this.



Look at all of the spaces, you have to be able to see all of this while just looking at the indentation. You don't have to just use spaces, you can also use tabs.

While Loops

A **while loop** runs as long as a specified condition is `True`. It's going to look inside the while loop at the indented lines of code to carry out those instructions, to do this, then do this, then do this. And finally, when it gets to the end of the while loop, it comes back to the beginning and tests the condition again And if it's still true, then it's going to go through and loop and loop and loop until this condition becomes false.

```python
number = 5
while number > 0:  # Keep running while the number is greater than 0
    print(number)
    number -= 1  # Reduce the number by 1 each time
```

While Loop vs For Loop

For Loop: Use when you know how many times to repeat.

While Loop: Use when you don't know how many times, but it depends on a condition.

```python
for i in range(3):  # Repeat 3 times
    print("Hello!")
```

```python
while not at_goal:  # Keep going until the robot reaches the goal
    jump()
```

Beware of Infinite Loops

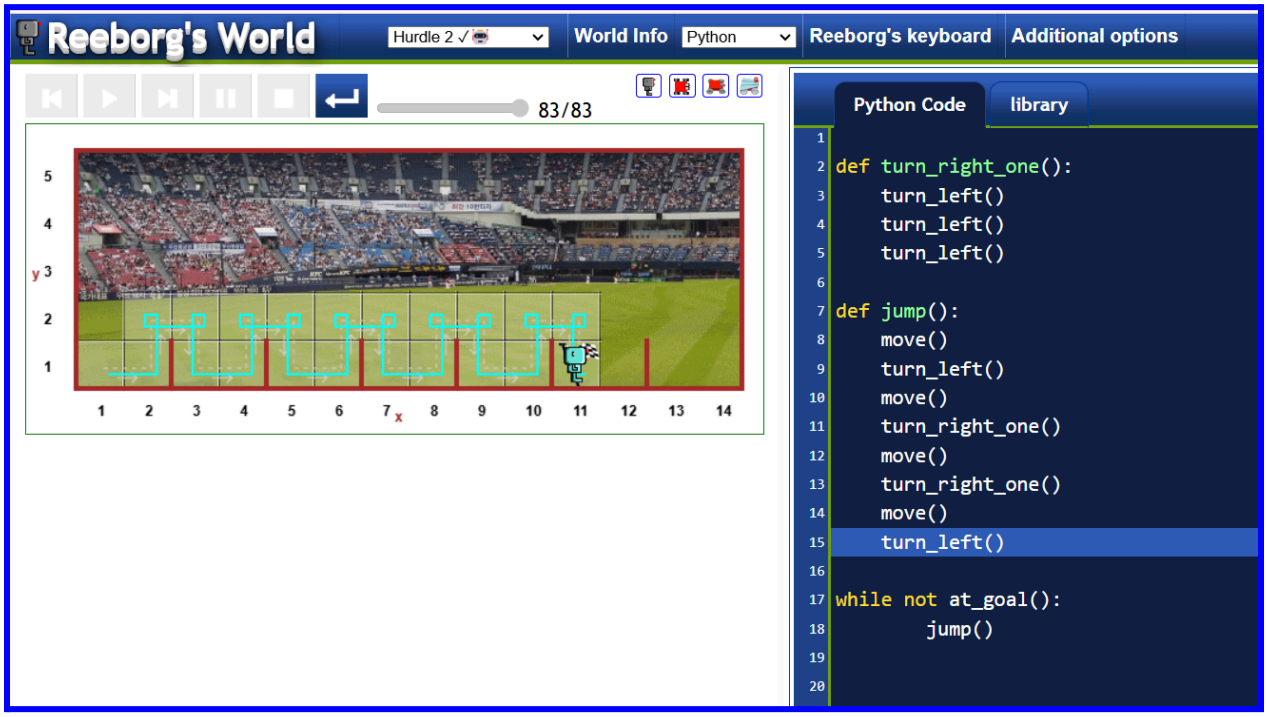
If the condition in a while loop never becomes false, it will run forever.

```python
while 5 > 3:  # This will never stop
    print("Help!")
```

When to Use Each?

- For Loop: When you know how many times to repeat.
- While Loop: When the loop should stop only when a condition changes (e.g., reaching a goal).

Challenge



Challenge #2