BASIC BUILT-IN FUNCTIONS

Function	Explanation
`print()`	Print text to the console.
`input()`	Get keyboard input via the console.
`import`	Announce the use of a specific library in our program.
`int()`	Convert a string to a numerical value of data type integer.
`float()`	Convert a string to a numerical value of data type float.
`...`	..
`...`	...

VARIABLES

Command	Explanation
`x = 1`	Declaration of a variable **x** with a value of type integer.
`PI = 3.14159`	Declaration of a constant PI with a value of type float.
`name = "Anna"`	Declaration of a variable with a value of type string.
`is_over_18 = True`	Declaration of a variable with a value of type boolean.
`ipcon = IPConnection()`	Declaration of a variable holding a complex object.

LOGICAL OPERATORS IN CONDITIONS

Operator	Explanation
`=`	Assign a value to a variable (not a logical operator!).
`==`	Compare two values and return **True** if they are equal.
`>` or `>=`	Compares two values and returns true if the left is greater than (or equal to) the right.
`<` or `<=`	Compares two values and returns true if the left is less than (or equal to) the right.
`!=`	Compares two values if they are not equal.
`and, or`	These operators logically combine two conditions.
`not`	Negates the expression's result.
`x is None`	Check if a value if **x** is **None**, i.e., not existing.

MATH OPERATORS

Operator	Explanation
`+ - / *`	The basic arithmetic operators.
`% //`	Calculates the modulo/integer division of two numbers.
`2**3`	Calculates 2 to the power of 3.
`pow(a, b)`	Calculates **a** to the power of **b**.

STRINGS

Expression	Explanation
`f"Pi is {PI}"`	Format string with placeholders in curly braces.
`name[1]`	Returns the second symbol from the string **name**.
`len(name)`	Returns the number of characters in the string.
`"n" in Anna`	Checks if the string on the left is contained in the string on the right.
`name.strip()`	Remove whitespaces at the beginning/end of a string.
`name.upper()`	Make all letters uppercase.
`name.lower()`	Make all letters lowercase.

TIME

Function	Explanation
`time.sleep(1.5)`	Wait for 1.5 seconds.
`time.time()`	Get the current date/time as a UNIX timestamp.

RANDOM

Function	Explanation
`random.randint(0, 10)`	Generate a pseudo-random whole number between 0 and 10.
`random.random()`	Generate a pseudo-random real number between 0 and 1.
`random.choice(list_a)`	Choose a random element from **`list_a`**, which must be a list/array.

MATH

Operator	Explanation
`math.sqrt(a)`	Calculate the square root from **a**.
`math.pi, math.e`	Get the value of the mathematical constants.
`round()`	Round to the nearest integer.
`math.floor()` `math.ceil()`	Round up or down.
`math.log()`	Natural logarithm
`math.log(x, base)`	Logarithm to base.

CONDITIONALS

Command	Explanation
`if x == 1:`	Check if **x** is one and execute the code block if true.
`elif x == 2:`	Add another case to an existing **`if`**-statement.
`else:`	Default code branch if all conditions are false.

CUSTOM FUNCTIONS

Operator	Explanation
`def my_func(x):`	Define a function **`my_func`** with one parameter **x**.
`return x**2`	Return a result from a function to the caller.
`def power(exp=2):`	Define a function with one parameter that has a default value.

LOOPS

Operator	Explanation
`while a < b:`	Repeat code as long as **a** is less than **b**.
`for i in range(10):`	Repeat 10 times (for values **`i`** = 0…9).
`break`	Leave the loop immediately.

LISTS

Expression	Explanation
`vals = [1, 2, 3]`	Create a list with three elements.
`vals[0]`	Get the first element from **`vals`**.
`vals.append(5)`	Append an element (5) to the list **`vals`**.
`vals[0:2]`	Get the first three elements from **`vals`**.
`len(vals9`	Get the number of elements in the list.
`vals.pop()`	Get the first element and remove it from **`vals`**.

`for v in vals:`
`  print(v)`
	Iterate through all elements in **`vals`** and store the current element on **`v`**. Print each element.