Course on Internet of Things

Exercises Session 1:

Exercise 1:

Connect to your ESP32 with minicom or thonny.

Using REPL:

- print “Hello World!”
- read in a text using *input()* and print it
- Calculate $127.9 * 157.6 \frac{17.2}{3.5*2^4}$
- Calculate sin(30°)
  If you see errors, how do you correct them? Are the results correct?

Exercise 2:

Write a script that assigns the values 5 and 3 to the variables a and b respectively

Print the results of 4 basic arithmetic operations:

- a + b
- a - b
- a * b
- a / b

If you use integer format when printing the result, you will see the result truncated to an integer. Can you correct changing the format?

Improve this program asking the user to enter 2 real numbers (floats) separated by a space. This is how it should look like:

```
MicroPython v1.12-421-g4371c971e on 2020-05-02; ESP32 module with ESP32
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
  Enter the calculation to be performed in the form: "operand1 operator operand2"
  Example: "5.3 + 4.2" or "1.4 * 7.9"
  Operation: 5.3 + 4.2
  5.3  +  4.2  =  9.5
>>>
```

If this happens:

```
>>> %Run -c $EDITOR_CONTENT

  Please enter two integer numbers separated by a space: d 6
  Traceback (most recent call last):
    File "<stdin>", line 24, in <module>
    File "<stdin>", line 17, in parse
  ValueError: invalid syntax for integer with base 10

>>>
```

when the user mistypes and enters something that is not a real number?

Can you capture the error and simply ask him to repeat his input until you get 2 correct floating-point numbers?

Bonus points: Writing a parser for user input is rather easy if the numbers and operator are separated by spaces (have a look at the split method of strings). It is quite a bit trickier if you allow entering calculations without spaces between the operands and the operator. Can you write a parser accomplishing this?

Hint: A very elegant solution to this is the use of regular expressions and Python provides a module helping you with this: https://docs.micropython.org/en/v1.14/library/ure.html

A float is defined as follows:

- zero or more spaces
- followed by one + or one -
- followed by
  - a '.' followed by at least one digit
- or
  - at least one digit followed by zero or one '.' followed by zero or more digits
- followed be zero of more spaces

Try your parser on this input:

Input	ok or not ok
5.3+4.7	ok
5.3++4.7	not ok
5..3 + 4.7	not ok
.3+.7	ok
5.3+4b7	not ok

empty string	not ok
5.3+.	not ok
5.3+	not ok
-1--1	ok
--1--1	not ok

Try also several spaces in between operands and operator.

Exercise 3:

Like in exercise2, start with 2 numbers a,b with values 5 and 7. In your program check which of the 2 numbers is bigger and print the result.

Then improve the program asking the user for 2 integer numbers. Make sure he enters 2 correct numbers and capture possible errors. This is a typical output:

```
>>> %Run -c $EDITOR_CONTENT

  Please enter two integer numbers separated by a space: 097403
  Please enter exactly 2 integer values
  Please enter two integer numbers separated by a space: 34h 23
  First value:  34h  is not an integer number, please repeat!
  Please enter two integer numbers separated by a space: 34 23
  The first value is bigger than the second
  because value 1 is 34 and value 2 is 23

>>>
```

Exercise 4: The Fibonacci series

The Fibonacci number series is defined as:

F(0) = 0; F(1) = 1

F(n) = F(n-1) + F(n-2)

- Calculate and print out the Fibonacci numbers up to F(n). Get n from the user where n >= 0

Try your program for n=0, n=1, n=2, n=3, n=20

- Ask the user up to which is the maximum number up to which you should calculate.

Exercise 5: A bit of Mathematics: Calculate the sine function

Write a script that asks the user for an angle in degrees. Calculate the sine of this angle.

Exercise 6: Classes

Write a Python class with methods calculating mathematical number series. These series should be calculated:

- The Fibonacci numbers up to F(n). n is passed as parameter to the method
- The Fibonacci numbers smaller than max. max is passed as parameter to the method
- The prime numbers up to max
- Factorial:

$$n! = 1 \cdot 2 \cdot 3 \cdots (n-2) \cdot (n-1) \cdot n,$$

- The geometric number series:

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots = \sum_{n=0}^{\infty} \frac{1}{2^n}.$$

- The harmonic number series:

$$\sum_{n=1}^{\infty} \frac{1}{n}.$$

Can you tell what the following series calculate?

$\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i}$	$\sum_{i=1}^{\infty} \frac{(-1)^{i+1}(4)}{2i-1}$	$2*(\frac{2*2}{1*3})*(\frac{4*4}{3*5})*(\frac{6*6}{5*7})*(\frac{8*8}{7*9})\ldots$	$\sum_{i=0}^{\infty} \frac{(-1)^{i}}{i!}$

Add these series to your class and try. Write a test script exercising the different methods.

In order to make this exercise run on the ESP32 you must first upload your mathSeries.py module implementing the MathSeries class into the file system of the ESP32. Create a directory called *lib* on the ESP32 file system and put your module there:

ampy mkdir /lib

ampy put mathSeries.py /lib/mathSeries.py

Once this is done, go to the REPL window of thonny and

import mathSeries

You can now create an instance of the MathSeries class:

series = mathSeries.MathSeries()

When running on a PC you may want to plot the series as well. If you don't know how to plot data in Python, you will find all the information needed here:

- https://matplotlib.org/3.2.1/index.html