Hello, world!

April 8, 2020

This is a first simple notebook showing how to print to console and how to read from it.

---

1 I/O to console

1.1 Print to console

Let's try to be polite

[9]:
```
print('Hello, world!')
```

```
Hello, world!
```

When printing to console, we can concatenate more sentences

[10]:
```
print('Hello, world!', 'What a beautiful day.', 'Indeed.')
```

```
Hello, world! What a beautiful day. Indeed.
```

1.2 Read from console

What if we want to send personal greetings!? :)

[12]:
```
print('What is your name?')
name=input()
print('Hello,', name)
```

```
What is your name?
```

```
 Andrea
```

```
Hello, Andrea
```

The read input is in a new line. How can I keep it in the same line? This solution offers a better result.

[13]:
```
name=input('What is your name? ')
print('Hello,', name)
```

```
What is your name?  Marco
```

```
Hello, Marco
```

[14]:
```
sign=input('What is your zodiac sign?')
```

```
What is your zodiac sign? Leo
```

[15]:
```
print('I foresee that in the coming months',sign,'will learn Python.')
```

```
I foresee that in the coming months Leo will learn Python.
```

2 Variables

Note that `name` and `sign` are **variables** that will store the read inputs.

[16]:
```
print('Hello again,', name)
```

```
Hello again, Marco
```

[17]:
```
print('Dear',name,'I foresee that in the coming months',sign,'will learn Python.
↪')
```

```
Dear Marco I foresee that in the coming months Leo will learn Python.
```

Every variable has a *name*. This name can start with a letter, a _,

[18]:
```
n=5
print(n)
```

```
5
```

[19]:
```
_n=5
print(_n)
```

```
5
```

Variables can contain other characters, but not as first one.

[20]:
```
n1=5
print(n1)
```

```
5
```

[21]:
```
1n=5
print(1n)
```

```
  File "<ipython-input-21-7a3b216150d8>", line 1
    1n=5
```

```
      ^
SyntaxError: invalid syntax
```

The values of variables can be updated.

[22]:
```
n=6
print(n)
```

```
6
```

We can even assign the value of a variable to another one

[26]:
```
print('Before:',n,n1)
n1=n
print('After',n,n1)
```

```
Before: 6 6
After 6 6
```

---

3 Next week...

Next week we will see that every variable has a *type*. Actually, everything has a type in Python. Everything is an *object* ...

[29]:
```
n1 = 5
n2 = 1.4e-2
n3 = 'Ciao'
n4 = n1 + n2
n5 = n1 + n3
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-29-90bbc4718efc> in <module>
         3 n3 = 'Ciao'
         4 n4 = n1 + n2
   ----> 5 n5 = n1 + n3


       TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

[ ]: