Variables, Values and Types

```
      #       The very basic use of a computer is to process data
      #       data can be anything: text, musik, an image, a website,
      #       data is used as input, as output and for internal processing
      #       all input, output and processing activity is handled by programs
      #       program languages use names to refer to data elements
      #       a program associates each data element with a certain 'type'

      #       So does Python

      #       Python knows integers, floating point numbers, strings and many more
      #       Values can be specified directly: this is called a 'literal'

V001 >>>      42   # a numeric literal with the value 42
     ==>      42
V002 >>>      3.14159   # a floating point literal with the value of PI (approx.)
     ==>      3.14159
V003 >>>      "Hello World"   #  a string literal
     ==>      'Hello World'
V004 >>>      False   # a boolean literal
     ==>      False
      #       the Python shell shows us each value
```

Literals and Simple Expressions

```
        #    Python can determine the type of a data element
V005 >>>     type(-99)         # for exaample: an integer
     ==>     <class 'int'>
V006 >>>     type(3.14159)     # each type has a name (of a class)
     ==>     <class 'float'>
V007 >>>     type('Lisboa')    # for now, let's ignore the 'class'-thing
     ==>     <class 'str'>
V008 >>>     type(True)
     ==>     <class 'bool'>

        #    Literals (like the above) can be used in expressions

V009 >>>     17 - 5
     ==>     12
V010 >>>     3 * 9.32
     ==>     27.96
V011 >>>     'hello' + ' ' + 'world'
     ==>     'hello world'
```

Variables

```
          #     a literal is a an example of data element
          #     to give a name to a data element we make an 'assignment'
V012 >>>        counter     # try, if this is a known name
     err!       NameError("name 'counter' is not defined",)
V013 >>>        counter = 77   # assign the value 77 to a name which is now a variable
V014 >>>        counter     # now display the value of the variable
     ==>        77
V015 >>>        counter = 162   # assign a different value
V016 >>>        counter     # the value has changed (thats why we call it a 'variable')
     ==>        162

          #     variables can be part of expressions
V017 >>>        counter - 98      # this does not change the variable
     ==>        64
V018 >>>        sum = 10 + counter / 2   # and expressions can be assigned to variables
V019 >>>        sum   # show the value of the new variable
     ==>        91.0

          #     its very important to understand the difference between an evaluation and an assignment
```

Rules for Literals and for Names

```
V020 >>>    3.9876   # floating point numbers are written with '.', not a comma
     ==>    3.9876
V021 >>>    'Lis' +  "boa"   # Strings are specifies with apostrophes or with double quotes
     ==>    'Lisboa'
       #    There is more to learn about strings and string literals, see the oots script for Strings

V022 >>>    name = 'Hans'
V023 >>>    Name = 'Diogo'
V024 >>>    NAME = 'Lara'
V025 >>>    print(name, Name, NAME)   # three different variables
     p()    Hans Diogo Lara
       #    the first letter of a name is '_' or 'a-z' or 'A-Z'
       #    all following letters can include also numbers '0-9'
       #    there is no length limit for a name
```

More about Assignment

```
V026 >>>    aaa = 'a_string'
V027 >>>    bbb = ccc = aaa   # chained assignment: both bbb and ccc get the same value as aaa
V028 >>>    print(aaa, bbb, ccc)
     p()    a_string a_string a_string
V029 >>>    aaa = aaa +'!'     # change the value of the variable
V030 >>>    print(aaa, bbb, ccc)
     p()    a_string! a_string a_string

       #    strings and numbers can never be changed, they are 'immutable'
       #    what we do is to assign a new value to the variable
```

More about Assignment

```
        #    To understand better, how Assignment works, we must use mutable types
V031 >>>     a = [1,2,3]    # assign a list (specified as a literal)
V032 >>>     b = [1,2,3]    # assign 'another' list (with the same value)
V033 >>>     a, b           # show the values
     ==>     ([1, 2, 3], [1, 2, 3])
V034 >>>     a[0] = 7       # change the 'a' list
V035 >>>     a, b           # b remains unchanged
     ==>     ([7, 2, 3], [1, 2, 3])
V036 >>>     a = b          # assign the value of b to a
V037 >>>     b[0] = 7       # change the 'b' list
V038 >>>     a, b           # watch out! There exists only one list, with two names
     ==>     ([7, 2, 3], [7, 2, 3])
V039 >>>     a = list(b)    # now we assign a copy of the 'b' list
V040 >>>     a[0] = 1       # change of a
V041 >>>     a, b           # ... leaves b untouched
     ==>     ([1, 2, 3], [7, 2, 3])

V042 >>>     a = b          # again
V043 >>>     del b          # this deletes only the name
V044 >>>     a              # the list remains untouched
     ==>     [7, 2, 3]
V045 >>>     b
     err!    NameError("name 'b' is not defined",)
```