```
Base Types
integer, float, boolean, string
   int 783
                        -192
                  0
float 9.23
                  0.0
                           -1.7e-6
                                   10-6
 bool True
                   False
   str "One\nTwo"
                             ' I\_',m '
             new line
                             ' escaped
                       """X\tY\tZ
              multiline
                      1\t2<u>\t</u>3"""
immutable.
ordered sequence of chars
                           tab char
```

```
Container Types

    ordered sequence, fast index access, repeatable values

                                              ["word"]
    list [1,5,9] ["x",11,8.9]
                                                               []
  tuple (1,5,9)
                          11, "y", 7.4
                                              ("word",)
                                                               ()
                      expression with just comas
immutable
     *str as an ordered sequence of chars
■ no a priori order, unique key, fast key access; keys = base types or tuples
    dict {"key":"value"}
                                                               {}
           {1: "one", 3: "three", 2: "two", 3.14: "π"}
key/value associations
     set {"key1", "key2"}
                                      {1,9,3,0}
                                                          set()
```

```
for variables, functions, modules, classes... names

a..zA..Z_ followed by a..zA..Z_0..9

diacritics allowed but should be avoided

language keywords forbidden

lower/UPPER case discrimination

a toto x7 y_max BigOne
```

⊗ 8y and

```
variables assignment

x = 1.2+8+sin(0)

value or computed expression
variable name (identifier)

y,z,r = 9.2,-7.6, "bad"

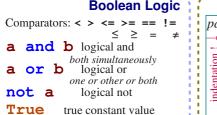
variables container with several values (here a tuple)

x+=3 increment decrement x-=2

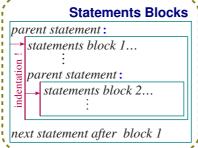
x=None «undefined» constant value
```

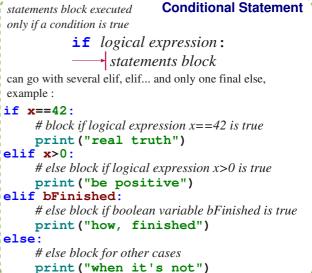
splitting string

```
for lists, tuples, strings, ... Sequences indexing
negative index | -6
                     -5
                                          -3
                                                   -2
                                                           -1
                                                                      len(lst) \longrightarrow 6
positive index 0
                     1
                               2
                                          3
                                                   4
                                                            5
                                                                    individual access to items via [index]
     lst=[11, 67,
                            "abc"
                                                  42;
                                        3.14,
                                                          1968]
                                                                      lst[1] \rightarrow 67
                                                                                                 1st [0] \rightarrow 11 first one
positive slice 0
                                                      5
                                                                      1st[-2] \rightarrow 42
                                                                                                 1st [-1] →1968 last one
negative slice -6 -5
                                               -2
                                                     -1
                        -4
                                     -¦3
                                                                    access to sub-sequences via [start slice:end slice:step]
     lst[:-1] \rightarrow [11, 67, "abc", 3.14, 42]
                                                                      lst[1:3] \rightarrow [67, "abc"]
     lst[1:-1] \rightarrow [67, "abc", 3.14, 42]
                                                                      lst[-3:-1] \rightarrow [3.14,42]
     lst[::2] \rightarrow [11, "abc", 42]
                                                                      lst[:3] \rightarrow [11, 67, "abc"]
     lst[:] \rightarrow [11, 67, "abc", 3.14, 42, 1968]
                                                                      lst[4:] \rightarrow [42, 1968]
                                       Missing slice indication \rightarrow from start / up to end.
         On mutable sequences, usable to remove del lst[3:5] and to modify with assignment lst[1:4]=['hop', 9]
```



False false constant value





```
Operators: + - * / / / % ** \\ \times \div \bigwedge \bigwedge a^b \\ \text{integer} \div \text{ remainder}
(1+5.3)*2 \rightarrow 12.6
abs (-3.2) \rightarrow 3.2
round (3.57, 1) \rightarrow 3.6
From math import sin, pi...
sin (pi/4) \rightarrow 0.707...
cos (2*pi/3) \rightarrow -0.4999...
acos (0.5) \rightarrow 1.0471...
sqrt (81) \rightarrow 9.0 \qquad \checkmark
log (e**2) \rightarrow 2.0 \qquad etc. (cf doc)
```

```
statements block executed as long Conditional loop statement \ i statements block executed for each
                                                                                                       Iterative loop statement
                                                                    item of a container or iterator
              while logical expression:
                                                                                     for variable in sequence:
                   → statements block
                                                             Loop control
                                                                                           ► statements block
 i = 1 initializations before the loop
                                                                immediate exit! Go over sequence's values
                                                                               s = "Some text"
 condition with at least one variable value (here i)
                                                                                                      \succ initializations before the loop
                                                                               cnt = 0
                                                                next iteration
 while i <= 100:
                                                                                 loop variable, value managed by for statement
       # statement executed as long as i \le 100
                                                                               for c in s:
                                                                                                                  Count number of
       s = s + i**2
                                                                                     if c == "e":
                                                                                                                  e in the string
       i = i + 1 } ⅓ make condition variable change
                                                                                          cnt = cnt + 1
                                                                               print("found", cnt, "'e'")
 print ("sum:", s) \rightarrow computed result after the loop
                                                                     loop on dict/set = loop on sequence of keys
                   🖆 be careful of inifinite loops!
                                                                     use slices to go over a subset of the sequence
                                                                     Go over sequence's index
                                               Display / Input
                                                                     □ modify item at index
                                                                     □ access items around index (before/after)
                                                                     lst = [11, 18, 9, 12, 23, 4, 17]
                                                                     lost = []
      items to display: litteral values, variables, expressions
                                                                     for idx in range(len(lst)):
    print options:
                                                                           val = lst[idx]
                                                                                                                Limit values greater
    □ sep=" " (items separator, default space)
                                                                           if val > 15:
                                                                                                                than 15, memorization
    □ end="\n" (end of print, default new line)
                                                                                                                of lost values.
                                                                                 lost.append(val)
    □ file=f (print to file, default standard output)
                                                                                 lst[idx] = 15
 s = input("Instructions:")
                                                                     print("modif:",lst,"-lost:",lost)
    input always returns a string, convert it to required type
                                                                     Go simultaneously over sequence's index and values:
                                                                     for idx, val in enumerate(lst):
       (cf boxed Conversions on on ther side).
'len (c) → items count
                                       Operations on containers
                                                                                                   Generator of int sequences
                                                                         frequently used in
                                                                                                                  not included
                                       Note: For dictionaries and set, these
                                                                         for iterative loops
min(c)
           max(c)
                        sum(c)
                                       operations use keys.
sorted (c) → sorted copy
                                                                                           range ([start,]stop [,step])
val in c → boolean, membersihp operator in (absence not in)

→ 0 1 2 3 4

                                                                         range (5)
enumerate (c) → iterator on (index,value)
                                                                         range (3, 8)
                                                                                                                  3 4 5 6 7
Special for sequence containeurs (lists, tuples, strings):
                                                                         range (2, 12, 3)-
                                                                                                                    2 5
reversed (\mathbf{c}) \rightarrow reverse iterator \mathbf{c} \star \mathbf{5} \rightarrow duplicate
                                                  c+c2 \rightarrow concatenate
c.index(val) → position
                                c.count (val) → events count
                                                                             range returns a « generator », converts it to list to see
                                                                             the values, example:
🕍 modify original list
                                               Operations on lists
                                                                             print(list(range(4)))
lst.append(item)
                                add item at end
lst.extend(seq)
                                add sequence of items at end
                                                                                                            Function definition
                                                                        function name (identifier)
!lst.insert(idx,val)
                                insert item at index
                                                                                               named parameters
lst.remove(val)
                                remove first item with value
lst.pop(idx)
                                remove item at index and return its value
                                                                         def fctname(p_x,p_y,p_z):
                                            sort / reverse list in place
lst.sort()
                  lst.reverse()
                                                                                """documentation"""
                                                                                # statements block, res computation, etc.
  Operations on dictionaries !
                                               Operations on sets
                                                                                return res ← result value of the call.
                                    Operators:
d[key]=value
                    d.clear()
                                    if no computed result to
d[key] \rightarrow value
                    del d[clé]
                                                                         parameters and all of this bloc
                                    & → intersection
                                                                                                       return: return None
                                                                         only exist in the block and during
d.update (d2) { update/add

    - ^ → difference/symetric diff

                                                                         the function call ("black box")
d.keys()
                  associations
                                    < <= > >= → inclusion relations
d.values() views on keys, values
                                    s.update(s2)
                                                                                                                   Function call
                                                                            = fctname(3,i+2,2*i)
d.items() | associations
                                    is.add(key) s.remove(key)
                                                                                             one argument per parameter
d.pop(clé)
                                    s.discard(key)
                                                                         retrieve returned result (if necessary)
 storing data on disk, and reading it back
                                                               Files
                                                                                                              Strings formating
   = open("fil.txt", "w", encoding="utf8")
                                                                          formating directives
                                                                                                         values to format
                                                                         "model {} {} {}".format(x,y,r) —
              name of file
                                                    encoding of
file variable
                              opening mode
                                                                         "{selection:formating!conversion}"
for operations on disk
                              □ 'r' read
                                                    chars for text
                                                                                               "{:+2.3f}".format(45.7273)
              (+path...)
                              □ 'w' write
                                                    files:
                                                                          Selection:
                              □ 'a' append...
                                                                                               →'+45.727'
                                                    11t f8
                                                            ascii
                                                                                              "{1:>10s}".format(8, "toto")
cf functions in modules os and os.path
                                                    latin1
                                                                           0.nom
                                                                                                          toto'
                                 empty string if end of file
    writing
                                                                           4[key]
                                                          reading
                                                                                               "{!r}".format("I'm")
                                                                           0 [2]
                                s = f.read(4)<sub>if char count not</sub>
f.write("hello")
                                                                                               →'"I\'m"'
                                                                        □ Formating :
                                     read next
                                                      specified, read
 fillchar alignment sign minwidth.precision~maxwidth type
                                                       whole file
 strings, convert from/to required
                                     line
 type. s = t.re
f.close() don't forget to close file after use
                                s = f.readline()
                                                                                 + - space
                                                                                             0 at start for filling with 0
                                                                        integer: b binary, c char, d decimal (default), o octal, x or X hexa...
                 Pythonic automatic close: with open (...) as f:
                                                                        float: e or E exponential, f or F fixed point, g or G appropriate (default),
 very common: iterative loop reading lines of a text file
                                                                               % percent
 for line in f :
                                                                        string: s ..
                                                                         □ Conversion: s (readable text) or r (litteral representation)
     d # line processing block
```