for variables, functions,

modules, classes... names

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

language keywords forbidden

diacritics allowed but should be avoided

```
License Creative Commons Attribution 4
                                      Base Types
integer, float, boolean, string, bytes
    int 783 0 -192
                             0b010 0o642 0xF3
 float 9.23 0.0
                              binary
                         -1.7<sub>e</sub>-6
                               \times 10^{-6}
  bool True False
     str "One\nTwo"
                               Multiline string:
                                  """X\tY\tZ
         escaped new line
                                  1\t2\t3"""
           'I<u>\</u>m'
           escaped '
                                    escaped tab
 bytes b"toto\xfe\775"
              hexadecimal octal
                                         ½ immutables
```

Identifiers

```
Container Types
• ordered sequences, fast index access, repeatable values
                                                                             []
          list [1,5,9]
                              ["x",11,8.9]
                                                          ["mot"]
       ,tuple (1,5,9)
                                  11, "y", 7.4
                                                          ("mot",)
                                                                              ()
 Non modifiable values (immutables)

    expression with only comas → tuple

                                                                             11/11
       *str bytes (ordered sequences of chars / bytes)
                                                                           b""
■ key containers, no a priori order, fast key access, each key is unique
dictionary dict {"key":"value"}
                                             dict (a=3,b=4,k="v")
                                                                             {;}
(key/value associations) {1:"one",3:"three",2:"two",3.14:"\pi"}
            set {"key1", "key2"}
                                             {1,9,3,0}
                                                                         set(i)

    ★ keys=hashable values (base types, immutables...)

                                             frozenset immutable set
                                                                            empty
```

```
□ lower/UPPER case discrimination

    □ a toto x7 y_max BigOne
    ○ 8y and for

                      Variables assignment
 assignment ⇔ binding of a name with a value
 1) evaluation of right side expression value
 2) assignment in order with left side names
x=1.2+8+\sin(y)
a=b=c=0 assignment to same value
y, z, r=9.2, -7.6, 0 multiple assignments
a,b=b, a values swap
a, *b=seq \ unpacking of sequence in
*a,b=seq \ item and list
                                                 and
             increment \Leftrightarrow x=x+3
x-=2
             decrement \Leftrightarrow \mathbf{x} = \mathbf{x} - \mathbf{2}
                                                  /=
x=None « undefined » constant value
del x
            remove name x
```

```
Conversions
                                              type (expression)
int("15") \rightarrow 15
int ("3f", 16) \rightarrow 63
                                   can specify integer number base in 2<sup>nd</sup> parameter
int(15.56) \rightarrow 15
                                   truncate decimal part
float ("-11.24e8") \rightarrow -1124000000.0
round (15.56, 1) \rightarrow 15.6
                                  rounding to 1 decimal (0 decimal \rightarrow integer number)
bool (x) False for null x, empty container x, None or False x; True for other x
str(x) \rightarrow "..." representation string of x for display (cf. formatting on the back)
\mathtt{chr}(64) \rightarrow '@'
                   ord('@')\rightarrow64
                                            code \leftrightarrow char
repr (\mathbf{x}) \rightarrow "..." literal representation string of \mathbf{x}
bytes([72,9,64]) \rightarrow b'H\t@'
list("abc") \rightarrow ['a', 'b', 'c']
\mathbf{dict}([(3,"three"),(1,"one")]) \rightarrow \{1:'one',3:'three'\}
set(["one","two"]) -> {'one','two'}
separator str and sequence of str \rightarrow assembled str
    ':'.join(['toto','12','pswd']) → 'toto:12:pswd'
str splitted on whitespaces \rightarrow list of str
    "words with spaces".split() → ['words','with','spaces']
str splitted on separator str \rightarrow list of str
   "1,4,8,2".split(",") \rightarrow ['1','4','8','2']
sequence of one type \rightarrow list of another type (via list comprehension)
    [int(x) for x in ('1', '29', '-3')] \rightarrow [1,29,-3]
```

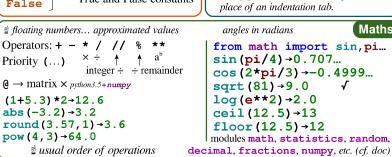
if a condition is true

if logical condition:

→ statements block

```
Sequence Containers Indexing
                                        for lists, tuples, strings, bytes...
                                          -2
                                                   -1
                                                                                     Individual access to items via lst [index]
                    -5
                           -4
                                   -3
                                                                Items count
                    0
                                    2
                                            3
                            1
                                                   4
   positive index
                                                            len(lst) \rightarrow 5
                                                                                     lst[0] \rightarrow 10
                                                                                                         ⇒ first one
                                                                                                                           lst[1] \rightarrow 20
          lst=[10,
                          20,
                                   30;
                                           40;
                                                   50]
                                                                                     1st [-1] → 50 \Rightarrow last one
                                                                                                                           lst [-2] \rightarrow 40
                                                               3
   positive slice
                                                                                      On mutable sequences (list), remove with
                                                             (here from 0 to 4)
                  -5
                               -3
                                       -2
                                              -1
   negative slice
                                                                                      del 1st[3] and modify with assignment
                                                                                      1st[4]=25
Access to sub-sequences via lst [start slice: end slice: step]
lst[:-1] \rightarrow [10, 20, 30, 40] lst[::-1] \rightarrow [50, 40, 30, 20, 10] lst[1:3] \rightarrow [20, 30]
                                                                                                                lst[:3] \rightarrow [10, 20, 30]
                                                                                 lst[-3:-1] \rightarrow [30,40] lst[3:] \rightarrow [40,50]
lst[1:-1] \rightarrow [20, 30, 40]
                                     lst[::-2] \rightarrow [50, 30, 10]
                                     1st[:]→[10,20,30,40,50] shallow copy of sequence
lst[::2] \rightarrow [10, 30, 50]
Missing slice indication \rightarrow from start / up to end.
On mutable sequences (1ist), remove with del 1st[3:5] and modify with assignment 1st[1:4]=[15,25]
```

```
Boolean Logic
                                                             Statements Blocks
 Comparisons : < > <= (boolean results) ≤
                           >= == !=
                                              parent statement :
                           ≥ =
                                               \rightarrow statement block 1...
 a and b logical and both simulta-
                            -neously
a or b logical or one or other or both
                                                  parent statement :
                                                     statement block2...
g pitfall: and and or return value of a or
of b (under shortcut evaluation).
\Rightarrow ensure that a and b are booleans.
                                               next statement after block 1
not a
                logical not
True
                                                d configure editor to insert 4 spaces in
                True and False constants
False
                                                place of an indentation tab.
```



```
module truc⇔file truc.py

from monmod import nom1, nom2 as fct

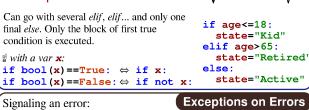
→direct access to names, renaming with as

import monmod →access via monmod.nom1 ...

modules and packages searched in python path (cf sys.path)

statement block executed only

Conditional Statement
```





```
Conditional Loop Statement | statements block executed for each | Iterative Loop Statement
   statements block executed as long as
                                                                                 item of a container or iterator
   condition is true
saool
                                                                                              for var in sequence:
      while logical condition:
                                                                        Loop Control
                                                                                                                                                 finish
infinite
                                                                                                   statements block
          \rightarrow statements block
                                                          break
                                                                         immediate exit
                                                          continue next iteration
                                                                                          Go over sequence's values
     = 0 \ initializations before the loop
                                                               beware of
                                                                                          s = "Some text" | initializations before the loop
  \mathbf{i} = \mathbf{1} condition with a least one variable value (here \mathbf{i})
                                                               loop exit.
                                                                                          cnt = 0
                                                                Algo:
   while i <= 100:
                                                                                                                                                   good habit : don't modify loop variable
                                                                      i = 100
                                                                                            loop, variable, assignment managed by for statement
                                                                       \sum_{i} i^2
        s = s + i**2
i = i + 1
                                                                                               c in s:
                                                                                          for
                                                                                               if c == "e":
                           🛮 make condition variable change!
                                                                                                                                  Algo: count
   print("sum:",s)
                                                                                                     cnt = cnt + 1
                                                                                                                                  number of e
                                                                                          print ("found", cnt, "'e'")
                                                                                                                                  in the string.
                                                                     Display
                                                                                 loop on dict/set ⇔ loop on keys sequences
 print("v=",3,"cm :",x,",
                                                                                 use slices to loop on a subset of a sequence
                                                                                 Go over sequence's index
      items to display: literal values, variables, expressions
                                                                                 □ modify item at index
 print options:
                                                                                 □ access items around index (before / after)
 □ sep="<sup>'</sup>"
                           items separator, default space
                                                                                 lst = [11, 18, 9, 12, 23, 4, 17]
 end="\n"
                           end of print, default new line
                                                                                 lost = []
 □ file=sys.stdout print to file, default standard output
                                                                                                                            Algo: limit values greater
                                                                                 for idx in range(len(lst)):
                                                                                       val = lst[idx]
                                                                                                                            than 15, memorizing
                                                                       Input
 s = input("Instructions:")
                                                                                       if val > 15:
                                                                                                                            of lost values.
                                                                                                                                                   521
                                                                                            lost.append(val)
    input always returns a string, convert it to required type
                                                                                            lst[idx] = 15
        (cf. boxed Conversions on the other side).
                                                                                 print("modif:", lst, "-lost:", lost)
                                    Generic Operations on Containers
len (c) \rightarrow items count
                                                                                 Go simultaneously over sequence's index and values:
min(c) max(c) sum(c)
                                             Note: For dictionaries and sets, these
                                                                                 for idx,val in enumerate(lst):
sorted(c) \rightarrow list sorted copy
                                              operations use keys.
val in c \rightarrow boolean, membership operator in (absence not in)
                                                                                                                             Integer Sequences
                                                                                   range ([start,] end [,step])
enumerate (c) \rightarrow iterator on (index, value)
                                                                                  zip(c1, c2...) \rightarrow iterator on tuples containing c_i items at same index
                                                                                  range (5) \rightarrow 0 1 2 3 4
                                                                                                               range (2, 12, 3) \rightarrow 25811
all (c) → True if all c items evaluated to true, else False
                                                                                  range (3,8) \rightarrow 3 4 5 6 7
                                                                                                               range (20, 5, -5) \rightarrow 20 15 10
any (c) \rightarrow True if at least one item of c evaluated true, else False
                                                                                  range (len (seq)) \rightarrow sequence of index of values in seq
                                                                                  a range provides an immutable sequence of int constructed as needed
Specific to ordered sequences containers (lists, tuples, strings, bytes...)
                                   c*5→ duplicate
reversed (c) \rightarrow inversed iterator
                                                         c+c2 \rightarrow concatenate
                                                                                                                              Function Definition
c.index (val) \rightarrow position
                                     c. count (val) \rightarrow events count
                                                                                  function name (identifier)
                                                                                            named parameters
import copy
copy.copy(c) → shallow copy of container
                                                                                   def fct(x, y, z):
                                                                                                                                            fct
copy . deepcopy (c) → deep copy of container
                                                                                         """documentation"""
                                                                                         # statements block, res computation, etc.
                                                      Operations on Lists
return res ← result value of the call, if no computed
lst.append(val)
                              add item at end
                                                                                                              result to return: return None
                              add sequence of items at end
lst.extend(seq)
                                                                                   parameters and all
lst.insert(idx, val)
                              insert item at index
                                                                                   variables of this block exist only in the block and during the function
                                                                                   call (think of a "black box")
lst.remove(val)
                              remove first item with value val
1st.pop ([idx]) \rightarrow value
                              remove & return item at index idx (default last)
                                                                                   Advanced: def fct(x,y,z,*args,a=3,b=5,**kwargs):
lst.sort() lst.reverse() sort / reverse liste in place
                                                                                    *args variable positional arguments (→tuple), default values,
                                                                                    **kwargs variable named arguments (→dict)
                                                      Operations on Sets
     Operations on Dictionaries
                                                                                   \mathbf{r} = \mathbf{fct}(3, \mathbf{i} + 2, 2 * \mathbf{i})
                                                                                                                                     Function Call
                                          Operators:
d[key] = value
                       d.clear()
                                           | → union (vertical bar char)
                                                                                   storage/use of
                                                                                                        one argument per
                       del d[key]
d[key] \rightarrow value
                                                                                   returned value
                                                                                                        parameter
                                           & → intersection
d.update (d2) { update/add associations

    - ^ difference/symmetric diff.

d.keys()
d.values()
d.items()
d.items()
                                                                                  this is the use of function
                                                                                                                Advanced:
                                                                                                                                              fct
                                           < <= >= \rightarrow inclusion relations
                                                                                 name with parentheses
                                                                                                                *seauence
                                          Operators also exist as methods.
                                                                                 which does the call
d.pop (key[,default]) \rightarrow value
                                          s.update(s2) s.copy()
                                                                                                                         Operations on Strings
                                                                                 s.startswith(prefix[,start[,end]])
                                          s.add(key) s.remove(key)
d.popitem() \rightarrow (key, value)
                                          s.discard(key) s.clear()
                                                                                 s.endswith(suffix[,start[,end]]) s.strip([chars])
d.get (key[,default]) \rightarrow value
                                                                                 s.count(sub[,start[,end]]) s.partition(sep) \rightarrow (before,sep,after)
                                          s.pop()
d.setdefault(key[,default]) \rightarrow value
                                                                                 s.index(sub[,start[,end]]) s.find(sub[,start[,end]])
                                                                       Files
                                                                                 s.is...() tests on chars categories (ex. s.isalpha())
 storing data on disk, and reading it back
                                                                                                                s.title() s.swapcase()
                                                                                                s.lower()
     f = open("file.txt", "w", encoding="utf8")
                                                                                 s.upper()
                                                                                 s.casefold()
                                                                                                    s.capitalize() s.center([width,fill])
               name of file
file variable
                                                                                 s.ljust([width,fill]) s.rjust([width,fill]) s.zfill([width])
                                  opening mode
                                                            encoding of
                                  □ 'r' read
□ 'w' write
for operations
               on disk
                                                            chars for text
                                                                                 s.encode (encoding)
                                                                                                          s.split([sep]) s.join(seq)
                (+path...)
                                                            files:
cf. modules os, os.path and pathlib ....'+' 'x' 'b' 't' latin1 ...
                                                                                                                  values to format
                                                                   ascii
                                                                                     formating directives
                                                                                                                                      Formatting
                                                                                   writing
                                 \begin{tabular}{l} $\sharp$ read empty string if end of file \end{tabular}
                                                                      reading
                                                                                   "{selection: formatting!conversion}"
 f.write("coucou")
                                 f.read([n])
                                                       \rightarrow next chars
                                                                                  □ Selection :
                                     if n not specified, read up to end!
                                                                                                              "{:+2.3f}".format(45.72793)
 f.writelines (list of lines)
                                 f.readlines ([n]) \rightarrow list of next lines f.readline() \rightarrow next line
                                                                                                              →'+45.728'
                                                                                     nom
                                                                                                              "{1:>10s}".format(8,"toto")
                                 f.readline()
                                                                                     0.nom
          text mode t by default (read/write str), possible binary
                                                                                                                        toto'
                                                                                     4 [key]
                                                                                                              "{x!r}".format(x="I'm")
          mode b (read/write bytes). Convert from/to required type!
                                                                                     0[2]
                                                                                                              →'"I\'m"'
f.close()
                     dont forget to close the file after use!
                                                                                   □ Formatting :
f.flush() write cache
                                    f.truncate([size]) resize
                                                                                  fill char alignment sign mini width precision~maxwidth type
                                                                                  <> ^= + - space
 reading/writing progress sequentially in the file, modifiable with:
                                                                                                          o at start for filling with 0
f.tell()\rightarrowposition
                                   f.seek (position[,origin])
                                                                                  integer: b binary, c char, d decimal (default), o octal, x or X hexa...
 Very common: opening with a guarded block
                                                with open (...) as f:
                                                                                  float: e or E exponential, f or F fixed point, f or f appropriate (default),
 (automatic closing) and reading loop on lines
                                                    for line in f :
                                                                                  string: s ...
                                                                                                                                    % percent
 of a text file:
                                                       # processing of line
                                                                                  □ Conversion: s (readable text) or r (literal representation)
```