Python 3 Cheat Sheet

Latest version on . https://perso.limsi.fr/pointal/python:memento

()

b""

{ }

set()

```
Base Types
                                                                                                            Container Types
integer, float, boolean, string, bytes
                                                  • ordered sequences, fast index access, repeatable values
                                                           list [1,5,9]
                                                                              ["x",11,8.9]
                                                                                                       ["mot"]
   int 783 0 -192
                          0b010 0o642 0xF3
float 9.23 0.0
                          binary
                                 octal
                                         hexa
                                                        ,tuple (1,5,9)
                                                                                 11, "y", 7.4
                                                                                                        ("mot",)
                      -1.7e-6
                                                  Non modifiable values (immutables)
                                                                                bool True False
                            ×10<sup>-6</sup>
                                                        * str bytes (ordered sequences of chars / bytes)
   str "One\nTwo"
                            Multiline string:
                                                  • key containers, no a priori order, fast key access, each key is unique
       escaped new line
                              """X\tY\tZ
                              1\t2\t3"""
                                                 dictionary dict {"key":"value"}
                                                                                            dict(a=3,b=4,k="v")
         'I<u>\</u>m'
         escaped '
                                                 (key/value associations) {1:"one", 3:"three", 2:"two", 3.14:"π"}
                                escaped tab
bytes b"toto\xfe\775"
                                                             set {"key1", "key2"}
                                                                                            {1,9,3,0}
            hexadecimal octal
                                     immutables

½ keys=hashable values (base types, immutables...)

                                                                                            frozenset immutable set
```

```
for variables, functions,
                                Identifiers
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
      8 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
x+=3
            increment \Leftrightarrow x=x+3
x - = 2
            decrement \Leftrightarrow \mathbf{x} = \mathbf{x} - \mathbf{2}
                                               /=
                                               용=
x=None « undefined » constant value
del x
            remove name x
```

```
empty
                                              type (expression)
                                                                           Conversions
int ("15") \rightarrow 15
                                   can specify integer number base in 2^{nd} parameter
int("3f",16) \rightarrow 63
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)
chr(64) \rightarrow '@' \quad ord('@') \rightarrow 64
                                            code \leftrightarrow char
repr (x) \rightarrow "..." literal representation string of x
bytes([72,9,64]) \rightarrow b'H\t@'
list("abc") \rightarrow ['a', 'b', 'c']
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() \rightarrow ['words','with','spaces']
\mathtt{str} splitted on separator \mathtt{str} \to \mathtt{list} of \mathtt{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]
```

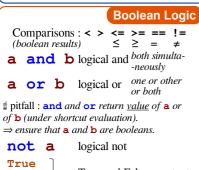
if a condition is true

if logical condition:

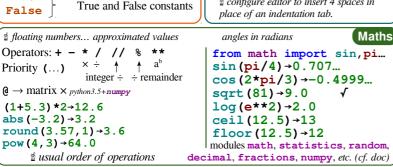
except Exception as e:

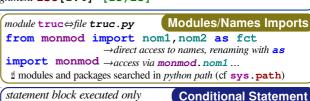
error processing block

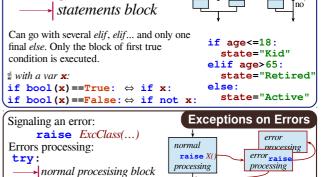
```
Sequence Containers Indexing
                                        for lists, tuples, strings, bytes...
                    -5
                           -4
                                    -3
                                          -2
                                                   -1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



```
Statements Blocks
parent statement :
 statement block 1...
   parent statement:
     statement block2...
next statement after block 1
 description configure editor to insert 4 spaces in
place of an indentation tab.
```







in all cases

finally block for final processing

```
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
infinite loops:
      while logical condition:
                                                                                               for var in sequence:
                                                                        Loop Control
                                                                                                                                                  finish
             statements block
                                                                          immediate exit
                                                                                                      statements block
                                                           break
                                                           continue next iteration
                                                                                            Go over sequence's values
   s = 0 initializations before the loop
                                                                ₫ else block for normal
ф
  i = 1 condition with a least one variable value (here i)
                                                                loop exit.
                                                                                           s = "Some text" initializations before the loop
beware
                                                                                           cnt = 0
                                                                 Algo:
                                                                                                                                                      good habit : don't modify loop variable
   while i <= 100:
                                                                       i = 100
                                                                                             loop variable, assignment managed by for statement or c in s:
                                                                       \sum_{i}^{2} i^{2}
        s = s + i**2
                                                                                           for
                           if c == "e":
        i = i + 1
  print("sum:",s)
                                                                                                      cnt = cnt + 1
                                                                                                                                    number of e
                                                                                           print("found", cnt, "'e'")
                                                                                                                                    in the string.
                                                                      Display
 print("v=", 3, "cm : ", x, ", ", y+4)
                                                                                   loop on dict/set ⇔ loop on keys sequences
                                                                                   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=" "
                            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.
                                                                                             lost.append(val)
    input always returns a string, convert it to required type
                                                                                   lst[idx] = 15
print("modif:",lst,"-lost:",lost)
        (cf. boxed Conversions on the other side).
len (c) → items count
                                    Generic Operations on Containers
                                                                                   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) → list sorted copy
                                              operations use keys.
val in c \rightarrow boolean, membership operator in (absence not in)
                                                                                                                               Integer Sequences
                                                                                     range ([start,] end [,step])
enumerate (\mathbf{c}) \rightarrow iterator on (index, value)
                                                                                   ₫ start default 0, end not included in sequence, step signed, default 1
zip (c1, c2...) \rightarrow iterator on tuples containing c<sub>i</sub> items at same index
                                                                                   range (5) \rightarrow 0 1 2 3 4
                                                                                                                 range (2, 12, 3) \rightarrow 25811
all (c) \rightarrow 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) → True if at least one item of c evaluated true, else False
                                                                                   range (len (seq)) \rightarrow sequence of index of values in seq
Specific to ordered sequences containers (lists, tuples, strings, bytes...)
                                                                                   arange provides an immutable sequence of int constructed as needed
reversed (c) \rightarrow inversed iterator c*5\rightarrow duplicate
                                                          c+c2→ concatenate
                                                                                                                                Function Definition
                                                                                   function name (identifier)
                                     c. count (val) \rightarrow events count
c.index (val) \rightarrow position
import copy
                                                                                                named parameters
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)
                                                                                    lst.insert(idx, val)
                               insert item at index
                                                                                    variables of this block exist only in the block and during the function
lst.remove(val)
                               remove first item with value val
                                                                                    call (think of a "black box")
                                                                                    Advanced: def fct(x,y,z,*args,a=3,b=5,**kwargs):
1st.pop ([idx]) \rightarrow value
                               remove & return item at index idx (default last)
lst.sort()
                 lst.reverse() sort / reverse liste in place
                                                                                      *args variable positional arguments (\rightarrow tuple), default values,
                                                                                      **kwargs variable named arguments (→dict)
     Operations on Dictionaries
                                                       Operations on Sets
                                           Operators:
                                                                                    \mathbf{r} = \mathbf{fct}(3, \mathbf{i} + 2, 2 * \mathbf{i})
                                                                                                                                       Function Call
                       d.clear()
d[key] = value
                                            I → 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.

                                                                                                                                                 fct
                                                                                   this is the use of function
                                                                                                                  Advanced:
                                            < <= > >= → inclusion relations
d.keys()
                                                                                   name with parentheses
                                                                                                                  *sequence
d.values() 

→ iterable views on 

d.items() 

keys/values/associations
                 →iterable views on
                                           Operators also exist as methods.
                                                                                   which does the call
                                                                                                                  **dict
                                           s.update(s2) s.copy()
d. pop (key[,default]) \rightarrow value
                                                                                                                           Operations on Strings
                                           s.add(key) s.remove(key)
                                                                                   s.startswith(prefix[,start[,end]])
d.popitem() \rightarrow (key, value) d.get(key[, default]) \rightarrow value
                                                                                   s.endswith(suffix[,start[,end]]) s.strip([chars])
                                           s.discard(key) s.clear()
                                           s.pop()
                                                                                   s.count(sub[,start[,end]]) s.partition(sep) \rightarrow (before,sep,after)
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.upper() s.lower()
                                                                                                                 s.title() s.swapcase()
     f = open("file.txt", "w", encoding="utf8")
                                                                                   s.casefold()
                                                                                                      s.capitalize() s.center([width,fill])
file variable
                name of file
                                   opening mode
                                                                                   s.ljust([width,fill]) s.rjust([width,fill]) s.zfill([width])
                                                            encoding of
for operations
                on disk
                                     'r' read
                                                            chars for text
                                                                                   s.encode (encoding)
                                                                                                            s.split([sep]) s.join(seq)
                                   □ 'w' write
                                                            files:
                (+path...)
cf. modules os, os.path and pathlib ....'+' 'x'
                                                                                      formating directives
                                                                                                                    values to format
                                                            utf8
                                                                    ascii
                                                                                                                                         Formatting
                                                 'b' 't' latin1 ...
                                                                                    "modele{} {} {}".format(x,y,r)—
                                  🖆 read empty string if end of file
                                                                       reading
                                                                                    "{selection: formatting!conversion}"
 f.write("coucou")
                                 f.read([n])
                                                        \rightarrow next chars
                                                                                   □ Selection :
                                                                                                                "{:+2.3f}".format(45.72793)
                                      if n not specified, read up to end!
 f.writelines (list of lines)
                                 f.readlines([n]) \rightarrow list of next lines
f.readline() \rightarrow next line
                                                                                      2
                                                                                                                →'+45.728'
                                                                                                               "{1:>10s}".format(8,"toto")

→' toto'
                                                                                      nom
                                 f.readline()
                                                                                      0.nom
           🖠 text mode t by default (read/write str), possible binary
                                                                                      4 [key]
                                                                                                                "{x!r}".format(x="I'm")
           mode b (read/write bytes). Convert from/to required type!
                                                                                      0[2]
                                                                                                               \rightarrow'"I\'m"'
f.close()
                     □ Formatting :
                                    f.truncate([size]) resize
f.flush() write cache
                                                                                    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() \rightarrow position
                                    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: \mathbf{e} or \mathbf{E} exponential, \mathbf{f} or \mathbf{F} fixed point, \mathbf{g} or \mathbf{G} appropriate (default),
 (automatic closing) and reading loop on lines
                                                     for line in f :
                                                                                    string: s ..
 of a text file:
                                                        # processing of line
                                                                                    □ Conversion: s (readable text) or r (literal representation)
```