Mémento v2.0.7

Python 3 Puska

Legutolsó verzió elérhetősége :
https://perso.limsi.fr/pointal/python:memento

Alaptípusok

egész,lebegőpontos,logikai,sztring,bájtok

```
int    783   0  -192   0b010 0o642 0xF3
                zéró   bináris oktális hexa-dec.
float  9.23  0.0  -1.7e-6
                          ×10^-6
bool   True  False
str    "Egy\nKettő"   Többsoros sztring:
       sztringben új sor """X\tY\tZ
       'I\'m'          1\t2\t3"""
       sztringben aposztróf  sztringben tabulátor
bytes  b"toto\xfe\775"
       hexadecimális oktális
```

nem módosíthatók (immutables)

Konténer típusok

- **sorozatok**, gyors hozzáférés index alapján, értékek ismétlődhetnek

```
list   [1,5,9]   ["x",11,8.9]   ["szó"]   []
tuple  (1,5,9)   11,"y",7.4     ("szó",)  ()
```

Értékek nem módosíthatók (immutables) *kifejezés mindössze veszőkkel elválasztva →* `tuple`

```
str bytes  (karakterek / bájtok sorozata)   ""  b""
```

- **kulcs alapú konténerek**, sorrend nem számít, gyors hozzáférés kulccsal, kulcs egyedi

```
szótár  dict  {"kulcs":"érték"}  dict(a=3,b=4,k="v")  {}
(kulcs/érték párok)  {1:"egy",3:"három",2:"kettő",3.14:"π"}
halmaz  set   {"kulcs1","kulcs2"}  {1,9,3,0}  set()
```

kulcsok=hash-elhető értékek (alaptípusok, immutables…) `frozenset` *immutable halmaz* — *üres*

Azonosítók

változó, függvény, modul, osztály... elnevezések

`a…zA…Z_` majd `a…zA…Z_0…9`

- ékezetek bár lehetségesek, de kerülendők
- a nyelv kulcsszavai nem használhatók
- kis/NAGY betűre érzékeny *(alma != Alma)*

☺ `a toto x7 y_max BigOne`
☹ ~~`8y and for`~~

Változó értékadás

`=`

értékadás ⇔ **összerendelése** névnek értékkel

1) kiértékelése a jobb odalon lévő kifejezésnek
2) értékadás a bal oldalon szereplő nevekhez

```
x=1.2+8+sin(y)
a=b=c=0   értékadás egyetlen értékkel több névhez
y,z,r=9.2,-7.6,0   többszörös értékadás
a,b=b,a   2 változó értékeinek felcserélése
a,*b=seq  }  sorozat kicsomagolása
*a,b=seq  }  értékbe és listába
x+=3   változó értékének növelése ⇔ x=x+3   (és *= /= %= …)
x-=2   változó érték csökkentése ⇔ x=x-2
x=None   « nem definiált » konstans érték
del x   változó eltávolítása
```

Konverziók

```
int("15") → 15          type(kifejezés)
int("3f",16) → 63       második paraméterben megadható a számrendszer bázisa
int(15.56) → 15         törtrész törlése
float("-11.24e8") → -1124000000.0
round(15.56,1)→15.6     kerekítés 1 tizedes jegyre (0 érték → egész szám)
bool(x)  False ha x zéró, x tartalma üres, x None vagy False; True más x értékre
str(x)→"…"   megjelenítési sztring az x értékre (formázott, olvashatóbb, mint a repr fv)
chr(64)→'@'  ord('@')→64   kód ↔ karakter
repr(x)→"…"  x érték tényleges karakteres formáját adja vissza
bytes([72,9,64]) → b'H\t@'
list("abc") → ['a','b','c']
dict([(3,"három"),(1,"egy")]) → {1:'egy',3:'három'}
set(["egy","kettő"]) → {'egy','kettő'}
elválasztó str és str elemek sorozata → összekonkatenált str
    ':'.join(['toto','12','jelszó']) → 'toto:12:jelszó'
str megvágva whitespace karaktereken → str list
    "szavak szóközökkel".split() → ['szavak','szóközökkel']
str elválasztva str karakteren → str list
    "1,4,8,2".split(",") → ['1','4','8','2']
sorozata adott típusnak → más típust tartalmazó list opcionális szűrőfeltétellel
 [int(x) for x in ('1','29','-3') if int(x) > 0 ] → [1,29]
```

Sorozat konténerek indexálása

listák, tuplék, sztringek és bájtok…

```
negatív index  -5  -4  -3  -2  -1
pozitív index   0   1   2   3   4
lst=[10,  20,  30,  40,  50]
pozitív szelet  0   1   2   3   4   5
negatív szelet -5  -4  -3  -2  -1
```

Elemek száma
`len(lst)→5`

index 0 -tól indul
(itt 0 -tól 4 -ig)

Hozzáférés adott elemhez `lst[`*index*`]`

`lst[0]→10` ⇒ első elem `lst[1]→20`
`lst[-1]→50` ⇒ utolsó elem `lst[-2]→40`

Módosítható sorozatokon (`list`), elem eltávolítása `del lst[3]` *és módosítása értékadással* `lst[4]=25`

Hozzáférés sorozat részéhez `lst[`*kezdő szelet*`:`*vége szelet*`:`*lépés*`]`

```
lst[:-1]→[10,20,30,40]   lst[::-1]→[50,40,30,20,10]   lst[1:3]→[20,30]   lst[:3]→[10,20,30]
lst[1:-1]→[20,30,40]     lst[::-2]→[50,30,10]         lst[-3:-1]→[30,40]  lst[3:]→[40,50]
lst[::2]→[10,30,50]      lst[:]→[10,20,30,40,50]   sekély klónozása (shallow copy) sorozatoknak
```

Hiányzó szelet megadás → kezdettől végig.
Módosítható sorozatokon (`list`), eltávolítás `del lst[3:5]` *és módosítás értékadással* `lst[1:4]=[15,25]`

Boole logika

Összehasonlítások: `<` `>` `<=` `>=` `==` `!=`
(logikai eredmények) ≤ ≥ = ≠

`a and b` és logika — *mindkettő egyszerre*

`a or b` vagy logika — *egyik, másik vagy mindkettő*

buktató : `and` *és* `or` *visszaadja* `a` *vagy* `b` *változók egyikének értékét, akkor is ha azok nem logikai értékeket tartalmaznak*

`not a` logikai nem

`True` `False` } igaz és hamis konstansok

Blokk utasítások

szülő utasítás `:`
→ *utasítás blokk #1…*
⋮
szülő utasítás `:`
→ *utasítás blokk #2…*
⋮
következő utasítás blokk #1 után

indentáció !

identációnak ajánlott **4** darab szóközt (tabulátor helyett)használni

Modulok/nevek importa

module `truc` ⇔ *fájl* `truc.py`

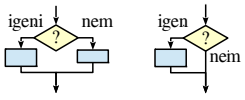
```
from monmod import nom1,nom2 as fct
    →direkt elérés nevekhez, átnevezéshez as
import monmod →hozzáférés monmod.nom1 …
```

modulok és packagek keresése *python path-on* (ld `sys.path`)

Feltételes utasítások

utasítás blokk végrehajtása csak **ha** *a feltétel igaz*

`if` *logikai feltétel* `:`
→ *utasítás blokk*



Kombinálható több **elif** feltétellel, de csak egyetlen végső **else** feltételt tartalmazhat. Csak az első igaz feltétel blokkja hajtódik végre.

```
if kor<=18:
  alp="Gyermek"
elif kor>65:
  alp="Nyugdíjas"
else:
  alp="Aktív"
```

`x` változó esetén :
`if bool(x)==True:` ⇔ `if x:`
`if bool(x)==False:` ⇔ `if not x:`

Matematika

lebegőpontos számok… közelítő értékek *szögek radiánban*

Műveletek : `+ - * / // % **`
Prioritás `(…)` × ÷ egész ÷ maradék ÷ a^b

`@` → mátrix × *python3.5+*`numpy`

```
(1+5.3)*2→12.6
abs(-3.2)→3.2
round(3.57,1)→3.6
pow(4,3)→64.0
```

szokásos sorrendje a műveleteknek

```
from math import sin,pi…
sin(pi/4)→0.707…
cos(2*pi/3)→-0.4999…
sqrt(81)→9.0   √
log(e**2)→2.0
ceil(12.5)→13
floor(12.5)→12
from random import randrange
randrange(kezdete,vége[,lépés])
```

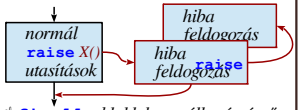
modulok `math`, `statistics`, `random`, `decimal`, `fractions`, `numpy`, stb.

Hiba kivételek

Hiba kiváltás / dobás :
`raise` *ExcClass(…)*

Hiba feldolgozás :

```
try:
→ normál utasítások blokkja
except ExcClass as e:
→ hiba feldolgozó blokk
```



`finally` *blokk használható végső utasítás blokknak – mindig lefut*

Feltételes ciklus utasítás

óvakodj a végtelen ciklusoktól!

utasítás blokk (ciklusé) végrehajtódik
amíg *a feltétel igaz*

```
while logikai feltétel:
    utasítás blokk
```


```
s = 0          } inicializálás a ciklus előtt
i = 1          }
while i <= 100:   } feltétel legalább a fenti változók egyikével (itt i)
    s = s + i**2
    i = i + 1
print("szumma:",s)
```

feltétel/ciklus változót módosítani kell, különben végtelen ciklus iterációt kapunk!

Algoritmus : $s = \sum_{i=1}^{i=100} i^2$

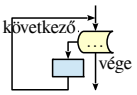
Ciklus vezérlés

```
break      azonnali kilépés
continue   következő iteráció
else utasítás blokk
```
normális ciklus kilépéskor

Iteratív ciklus utasítás

utasítás blokk (ciklusé) végrehajtódik
minden *elemére a konténernek vagy iterátornak*

```
for var in sorozat:
    utasítás blokk
```



Végiglépkedés sorozat értékein

```
s = "Eme lelet"   } inicializálás a ciklus előtt
cpt = 0
for c in s:
    if c == "e":
        cpt = cpt + 1
print("talált",cpt,"'e'")
```

ciklus változó, a tényleges értékadást a for utasítás végzi

Algoritmus : megszámlálja az **e** *karakter előfordulását a sztringben.*

ciklus szótáron/halmazon ⇔ ciklus a kulcsok sorozatán
használj szeleteket a ciklusban, hogy alsorozatokon menj végig

Végiglépkedés konténer sorozat indexein
- elem módosítása pozíción
- elemek elérése pozíción (előtte/utána)

```
lst = [11,18,9,12,23,4,17]
torolt = []
for idx in range(len(lst)):
    ertek = lst[idx]
    if ertek > 15:
        torolt.append(ertek)
        lst[idx] = 15
print("módosított:",lst,"-elvesztett:",elvesztett)
```

Algoritmus: limitálja a 15-nél nagyobb értékeket egy listában, eltárolva az elveszett értékeket egy új listában.

jó tanács : ne módosítsd a ciklus változót

Végiglépkedés sorozat indexein és értékein egyszerre

```
for idx,ertek in enumerate(lst):
```

Megjelenítés

```
print("v=",3,"cm :",x,",",y+4)
```

megjelenítendő elemek : literál értékek, változók, kifejezések

print opciói:
- **sep**=" " elválasztó karakter elemek között, alapból szóköz
- **end**="\n" sor lezáró karakter, alapból sorvége
- **file**=sys.stdout kiiratás fájlba, alapból standard kimenetre

Bemenet

```
s = input("Utasítások:")
```

input mindig egy **sztring**-et ad vissza, konvertálandó a kívánt típusra (lásd Konverziók rész a másik odalon).

Generikus műveletek konténereken

```
len(c) → elemek száma
min(c)  max(c)  sum(c)
sorted(c) → list rendezett másolat
ertek in c → logikai, tartalmazza teszt in (nem tartalmazza not in)
enumerate(c) → iterátor (index, érték)
zip(c1,c2…) → iterátor tuplékon amelyek ci értékeit tartalmazzák egyazon indexen
all(c) → True ha minden eleme c-nek igaz-ra értékelődik, különben False
any(c) → True ha legalább egy eleme c-nek igaz-ra értékelődik, különben False
c.clear() törli az értékeit a szótáraknak, halmazoknak, listáknak
```

Megj.: Szótárakon és halmazokon ezek a műveletek a kulcsokat használják.

*Specifikus **rendezett sorozatokhoz** (listák, tuplék, sztringek, bájtok…)*

```
reversed(c) → inverz iteráció     c*5→ multiplikáció   c+c2→ konkatenáció
c.index(ertek) → pozíció          c.count(ertek) → előfordulások száma
```

```
import copy
copy.copy(c) → sekély másolata (shallow copy) a konténernek
copy.deepcopy(c) → mély másolata (deep copy) a konténernek
```

Műveletek listákon

eredeti lista módosul

```
lst.append(ertek)      ertek elem hozzáadása a végéhez
lst.extend(sorozat)    elemek sorozat hozzáadása a végéhez
lst.insert(idx,ertek)  ertek elem beszúrása idx pozíción
lst.remove(ertek)      eltávolítása az ertek elem első előfordulásának
lst.pop([idx]) →érték  eltáv. & lekérése elemnek idx pozíción (alapból utolsót)
lst.sort()  lst.reverse()  rendezés / inverzió helyben
```

Műveletek szótárakon

```
d[kulcs]=érték      del d[kulcs]
d[kulcs] → érték
d.update(d2)   módosít / hozzáad összerendeléseket
d.keys()   }
d.values() } →iteráció kulcsokon, értékeken és párokon
d.items()  }
d.pop(kulcs[,alapérték]) → érték
d.popitem() → (kulcs,érték)
d.get(kulcs[,alapérték]) → érték
d.setdefault(kulcs[,alapért]) →érték
```

Műveletek halmazokon

Műveletek :
- | → unió
- & → (közös)metszet
- – ^ → különbség/szimmetrikus kül.
- < <= > >= → belefoglalás relációk

Fenti műveletek szintén léteznek halmaz függvényekként is.

```
s.update(s2)  s.copy()
s.add(kulcs)  s.remove(kulcs)
s.discard(kulcs)  s.pop()
```

Fájlok

adatok tárolása szekvenciális módon és azok visszaolvasása

```
f = open("fic.txt","w",encoding="utf8")
```

változó fájl változó továb-(+path…) bi műveletekhez

fájlnév

ld. modulok **os**, **os.path** és **pathlib**

megnyitási **mód**
- 'r' olvasás (read)
- 'w' írás (write)
- 'a' bővítés (append)
- …'+' 'x' 'b' 't'

karakter **kódolás** szöveg fájlokhoz:
utf8 ascii latin1 …

írás

```
f.write("kakukk")
f.writelines(sorok listája)
```

olvasás

üres sztringet olvas be a fájl végén

```
f.read([n])       → következő karakterek
                    if nincs n megadva, a fájl végéig olvas !
f.readlines([n])  → következő sorok listája
f.readline()      → következő sor
```

szöveges mód **t** *alapból (olvas/ír* **str***), bináris mód* **b** *lehetséges (olvast/ír* **bytes***). **Konvertálj a megfelelő típusra** !*

```
f.close()
```
ne felejtsd **lezárni a fájlt** használat után !

```
f.flush()  cache kiiratása/ürítése   f.truncate([taille])  újraméretezés
```

olvasás/írás szekvenciális módon történik a fálba, de az aktuális pozíció módosítható

```
f.tell() →pozíció        f.seek(posíció[,eredeti])
```

Nagyon gyakori : szöveg fájl megnyitás kontrollált blokkal (automatikus fájl lezárással) és beolvasása a soroknak ciklusban.

```
with open(…) as f:
    for sor in f :
        # sor feldolgozása
```

Egész sorozatok

```
range([kezdete,] vége [,lépés])
```
kezdete alapból 0, *vége* nincs benne a sorozatban, *lépés* előjeles és alapból 1

```
range(5) → 0 1 2 3 4          range(2,12,3) → 2 5 8 11
range(3,8) → 3 4 5 6 7        range(20,5,-5) → 20 15 10
range(len(sorozat)) → sorozat sorozatot alkotó index-ek sorozata
```
range nem módosítható sorozatot tartalmaz meghatározott egész számokból

Függvény definíció

függvény neve (azonosító)
paraméter nevek

```
def fv(x,y,z):
    """dokumentáció"""
    # utasítás blokk, eredmény számítás, stb.
    return res
```

eredmény értéke a hívásnak, ha nem szerepel **return None** hajtódik végre

a paraméterei és az összes változója a blokknak csak a blokkon belül létezik és csak a függvény hívás időtartama alatt (« fekete doboz »)

Haladó : `def fv(x,y,z,*args,a=3,b=5,**kwargs):`

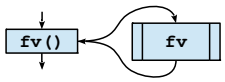
args változó pozíciójú/számú argumentumok (→tuple**), alapértékkel rendelkezők, **kwargs változó elnevezésű argumentumok (→**dict**)*

Függvény hívás

```
r = fv(3,i+2,2*i)
```
tárolása/használata a visszatérési értéknek
egy argumentum érték paraméterenként

a függvény hívásához a nevét kell megadni, zárójelek között pedig a paramétereket, ha léteznek

Haladó:
**sorozat*
***szótár*



Sztring műveletek

```
s.startswith(prefix[,kezdete[,vége]])  s.strip([karakterek])
s.endswith(suffix[,kezdete[,vége]])    s.partition(elv) → (előtte,elv,utána)
s.count(sub[,kezdete[,vége]])          s.center([szél,str])
s.index(sub[,kezdete[,vége]])          s.find(sub[,kezdete[,vége]])
s.is…()  különböző sztring tesztek (pl. s.isalpha())
s.upper()  s.lower()  s.title()  s.swapcase()
s.casefold()  s.capitalize()  s.center([szél,str])
s.ljust([szél,str])  s.rjust([szél,str])  s.zfill([szélesség])
s.encode(kódolás)  s.split([elválasztó])  s.join(sorozat)
```

Formázás

formázó direktívák értékek a formázáshoz

```
"modell{} {} {}".format(x,y,r) → str
"{szelekció:formázás!konverzió}"
```

- **Szelekció** :
  - 2
  - név
  - 0.név
  - 4[kulcs]
  - 0[2]

Példák:
```
"{:+2.3f}".format(45.72793)
→'+45.728'
"{1:>10s}".format(8,"toto")
→'      toto'
"{x!r}".format(x="L'ame")
→'"L\'ame"'
```

- **Formázás** :

kitölt.kar. *igazítás* *előjel* *min.szél.* *.pontosság~max.szél.* *típus*

`< > ^ =` `+ – szóköz` `0` kezdetben töltse fel 0 értékekkel

egész : **b** bináris, **c** karakter, **d** decimal (alapérték), **o** oktális, **x** vagy **X** hexa lebegőp. : **e** vagy **E** exponenciális, **f** vagy **F** fixpontos, **g** vagy **G** megfelelően (alapérték), **%** százalék
sztring : **s** …

- **Konverzió** : **s** (olvasható szöveg) vagy **r** (szószerinti reprezentáció)