

integer, float, boolean, string, bytes

Typy proste

int 783 0 -192 0b010 0o642 0xF3
l. całkowita zero binarny ósemkowy heksadecymalny
float 9.23 0.0 -1.7e-6
l. zmiennoprzecinkowa typ logiczny
bool True False
typ logiczny
str "One\nTwo"
łańcuch znaków
Przejdź do nowej linii
bytes b"toto\xfe\x775"
bajt
szesnastkowy ósemkowy
Wieloletniowy string:
str "X\ty\tz
1\t2\t3"
Wyświetlenie znaku ' Wyświetlenie tabulacji
niezmienne

• uporządkowane sekwencje, szybki dostęp do indeksów, powtarzalne wartości

Rodzaje kontenerów

list [1,5,9] **tuple** (1,5,9) **dict** {"key": "value"} **set** {"key1", "key2"}
Wartości niemodyfikowalne (niezmienne)
wyrażenie z tylko przecinkami → tuple
(uporządkowane sekwencje znaków / bajtów)
dict {"key": "value"} **dict** (a=3, b=4, k="v")
(pary klucz/wartość)
set {"key1", "key2"} **set** {1, 9, 3, 0}
klucze = wartości, które można mieszać (typy podstawowe, niezmienne, itd) frozenset niezmieniany zestaw pusty

dla nazw zmiennych,
funkcji, modułów, klas ...

Identyfikatory

a...zA...Z po którym mogą być **a...zA...Z_0...9**
• znaki diakrytyczne dozwolone, ale należy ich unikać
• zabronione słowa kluczowe w języku
• rozróżnianie małych/DUŻYCH liter
• **a toto x7 y_max BigOne**
• **8y and for**

= Przypisanie do zmiennych

• zadanie ⇔ wiązanie nazwy z wartością
1) ocena wartości wyrażenia po prawej stronie
2) przypisanie w kolejności z nazwami po lewej stronie
x=1.2+8+sin(y)
a=b=c=0 przypisanie do tej samej wartości
y,z,r=9.2,-7.6,0 równoczesne przypisanie
a,b=b,a zamiana wartości
a,*b=seq sekwencja rozpakowywania w
***a,b=seq** pozycji i w listach
x+=3 inkrementacja ⇔ **x=x+3**
x-=2 dekrementacja ⇔ **x=x-2**
x=None «niezdefiniowany» stała wartość
del x usuwa nazwę **x**

int ("15") → 15
int ("3f", 16) → 63
int (15.56) → 15
float ("-11.24e8") → -1124000000.0
round (15.56, 1) → 15.6
bool (x) False dla zera x, pusty kontener x, None lub False x; True dla innych x
str (x) → "..." reprezentuje łańcuch x do wyświetlenia (por. formatowanie z tyłu)
chr (64) → '@' **ord** ('@') → 64 kod ↔ znak
repr (x) → "..." dosłowny ciąg reprezentujący x
bytes ([72, 9, 64]) → b'H\t@'
list ("abc") → ['a', 'b', 'c']
dict ([(3, "three"), (1, "one")]) → {1: 'one', 3: 'three'}
set (["one", "two"]) → {'one', 'two'}
separator str i sekwencja str → złożony **str**
':'.join(['toto', '12', 'pswd']) → 'toto:12:pswd'
str wycięcie białych znaków → **list** utworzona z **str**
"words with spaces".split() → ['words', 'with', 'spaces']
str rozdzielanie na separatorze **str** → **list** utworzona z **str**
"1,4,8,2".split(",") → ['1', '4', '8', '2']
sekwencja jednego typu → **list** innego typu
[int(x) for x in ('1', '29', '-3')] → [1, 29, -3]

type (expression)

Konwersje typów

dla list, krotek, łańcuchów znaków, bajtów...

Indeksowanie kontenerów sekwencji

indeks ujemny -5 -4 -3 -2 -1
indeks dodatni 0 1 2 3 4
lst=[10, 20, 30, 40, 50]
wycinek dodatni 0 1 2 3 4 5
wycinek ujemny -5 -4 -3 -2 -1

Liczba elementów
len (lst) → 5
index od 0
(tutaj od 0 do 4)

Indywidualny dostęp do elementów przez **lst[index]**
lst[0]→10 ⇒ pierwszy **lst[1]→20**
lst[-1]→50 ⇒ ostatni **lst[-2]→40**
Na modyfikowalnych sekwencjach (**list**),
usuwanie za pomocą **del lst[3]** i
według przypisania **lst[4]=25**

Dostęp do sekwencji podrzędnych za pośrednictwem **lst[start slice: end slice: step]**

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]** płytka kopia sekwencji

Brak wskazania wycinka → od początku / do końca.

W sekwencjach mutowalnych (lista) (**list**), usuń za pomocą **del lst[3:5]** i modyfikuj z przydziałem **lst[1:4]=[15,25]**

Logika Boole'a

Bloki instrukcji

moduł **truc** ⇔ plik **truc.py**

Importy modułów / nazw

Porównania: < > <= >= == !=
(wyniki logiczne)
a and b logiczne i oba jednocześnie
a or b logiczne lub jeden lub inny lub oba
• pułapka: **and** i **or** zwraca wartość a lub b ("short-circuit" operator)
⇒ upewnij się, że a i b są wartościami logicznymi.
not a logiczne nie
True
False } stałe prawda i fałsz

blok rodzica:
blok instrukcji 1...
blok instrukcji 2...
następna instrukcja po bloku 1
skonfiguruj edytor tak, aby wstawiał 4 spacje w miejsce wcięcia.

from monmod import nom1, nom2 as fct
→ bezpośredni dostęp do nazw; zmiana nazwy za pomocą **as**
import monmod → dostęp przez **monmod.nom1** ...
• moduły i pakiety przeszukane w ścieżce **python path** (por **sys.path**)

blok instrukcji wykonywany tylko
jeśli warunek jest prawdziwy

Instrukcje warunkowe

if warunek logiczny:
blok instrukcji

tak nie
tak nie

Może iść z kilkoma **elif**, **elif...** i tylko jednym
kończącym **else**. Wykonywany jest tylko blok
pierwszego prawdziwego warunku.

if age<=18:
state="Dziecko"
elif age>65:
state="Emeryt"
else:
state="Pracownik"

z **var x:**

if bool(x)==True: ⇔ **if x:**
if bool(x)==False: ⇔ **if not x:**

Sygnalizacja błędów:

raise ExcClass(...)

Przetwarzanie błędów:

try:
normalny blok przetwarzania

except Exception as e:
blok przetwarzania błędów

Wyjątki
normalne przetwarzanie
błąd przetwarzania
błąd **raise** przetwarzania
finally blok do ostatecznego
przetworzenia we wszystkich przypadkach

liczby zmiennoprzecinkowe... wartości przybliżone

Operatory: + - * / // % **
Priorytet (...)
l. całkowita ÷ reszta
@ → macierz × 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
zwykła kolejność operacji

kąty w radianach

from math import sin, pi...
sin(pi/4)→0.707...
cos(2*pi/3)→-0.4999...
sqrt(81)→9.0
log(e2)→2.0**
ceil(12.5)→13
floor(12.5)→12

Matematyka

moduły **math**, **statistics**, **random**,
decimal, **fractions**, **numpy**, itp. (por. doc)

