Vestavěné funkce

A
abs()
aiter()
all()
any()
anext()
ascii()

B
bin()
bool()
breakpoint()
bytearray()
bytes()

C
callable()
chr()
classmethod()
compile()
complex()

D
delattr()
dict()
dir()
divmod()

E
enumerate()
eval()
exec()

F
filter()
float()
format()
frozenset()

G
getattr()
globals()

H
hasattr()
hash()
help()
hex()

I
id()
input()
int()
isinstance()
issubclass()
iter()

L
len()
list()
locals()

M
map()
max()
memoryview()
min()

N
next()

O
object()
oct()
open()
ord()

P
pow()
print()
property()

R
range()
repr()
reversed()
round()

S
set()
setattr()
slice()
sorted()
staticmethod()
str()
sum()
super()

T
tuple()
type()

V
vars()

Z
zip()

_
__import__()

abs(x)

Vrátí absolutní hodnotu čísla.

aiter(async_iterable)

Vraťte asynchronní iterátor pro asynchronní iterovatelný .

all(iterable)

Vrátit se True pokud jsou všechny prvky iterovatelné

awaitable anext(async_iterator)
awaitable anext(async_iterator, default)

Při čekání vraťte další položku z daného asynchronního iterator , nebo *výchozí,* pokud je zadán a iterátor je vyčerpán.

any(iterable)

Vrátit se True je-li některý prvek iterovatelného *pravdivý* .

ascii(object)

Tak jako repr(), vrátí řetězec obsahující tisknutelnou reprezentaci an objekt 2.

bin(x)

Převeďte celé číslo na binární řetězec s předponou „0b“.

class bool(x=False)

Vrátí booleovskou hodnotu, tj. jednu z True nebo False

breakpoint(*args, **kws)

Tato funkce vás přenese do ladicího programu na místě volání

class bytearray(source=b'')
class bytearray(source, encoding)
class bytearray(source, encoding, errors)

Vraťte nové pole bajtů. The bytearraytřída je proměnlivá posloupnost celých čísel v rozsahu 0 <= x < 256.

class bytes(source=b'')
class bytes(source, encoding)
class bytes(source, encoding, errors)

Vrátí nový objekt „bytes“, který je neměnnou posloupností celých čísel rozsah 0 <= x < 256

callable(object)

Vrátit se True pokud *se argument objektu* zdá být volatelný, False Pokud ne.

chr(i)

Vrátí řetězec představující znak, jehož kód Unicode je bod celé číslo *i*

@classmethod

Transformujte metodu na metodu třídy.

compile(source, filename, mode, flags=0, dont_inherit=False, optimize=- 1)

Zkompilujte *zdroj* do kódu nebo objektu AST.

class complex(real=0, imag=0)
class complex(string)

Vraťte komplexní číslo s hodnotou *real* + *imag* *1j nebo převeďte řetězec nebo číslo na komplexní číslo.

delattr(object, name)

Toto je příbuzný setattr(). Argumenty jsou objekt a a tětiva.

class dict(kwarg)**
class dict(mapping, **kwarg)
class dict(iterable, **kwarg)

Vytvořte nový slovník. The dictobjekt je třída slovníku.

dir()
dir(object)

Bez argumentů vraťte seznam názvů v aktuálním místním oboru. s argument, pokuste se vrátit seznam platných atributů pro daný objekt.

divmod(a, b)

Vezměte dvě (nekomplexní) čísla jako argumenty a vraťte dvojici čísel skládající se z jejich podílu a zbytku při použití celočíselného dělení.

enumerate(iterable, start=0)

Vraťte objekt výčtu. *iterovatelný*

eval(expression, globals=None, locals=None)

Argumenty jsou řetězec a volitelné globals a locals.

exec(object, globals=None, locals=None, /, *, closure=None)

Tato funkce podporuje dynamické provádění kódu Pythonu. *objekt* musí být buď řetězec nebo objekt kódu.

filter(function, iterable)

Sestavte iterátor z těch prvků iterovatelné *pro* kterou *funkci* vrátí true.

class float(x=0.0)

Vrátí číslo s plovoucí desetinnou čárkou vytvořené z čísla nebo řetězce *x* .

format(value, format_spec='')

Převeďte *hodnotu* na „formátovanou“ reprezentaci, kterou řídí *format_spec* .

class frozenset(iterable=set())

Vraťte nový frozensetobjekt, volitelně s prvky převzatými z *iterovatelný* .

getattr(object, name)
getattr(object, name, default)

Vrátí hodnotu pojmenovaného atributu *objektu* . *jméno* musí být řetězec.

globals()

Vraťte slovník implementující aktuální jmenný prostor modulu.

hasattr(object, name)

Argumenty jsou objekt a řetězec. Výsledek je Truepokud řetězec je název jednoho z atributů objektu, False Pokud ne.

hash(object)

Vraťte hash hodnotu objektu (pokud nějakou má).

help()
help(request)

Vyvolejte vestavěný systém nápovědy

hex(x)

Převeďte celé číslo na malý hexadecimální řetězec s předponou "0x".

id(object)

Vraťte „identitu“ objektu.

input()

input(prompt)

Pokud *je přítomen argument prompt* , je zapsán na standardní výstup bez koncový nový řádek. Funkce pak načte řádek ze vstupu, převede jej na řetězec (odstranění koncového nového řádku) a vrátí to.

class int(x=0)

class int(x, base=10)

Vrátí celočíselný objekt vytvořený z čísla nebo řetězce *x* nebo return 0pokud nejsou uvedeny žádné argumenty.

isinstance(object, classinfo)

Vrátit se True pokud *je argument object* instancí třídy *classinfo* argument nebo jeho (přímé, nepřímé nebo virtuální) podtřídy.

issubclass(class, classinfo)

Vrátit se Trueif *class* je podtřída (přímá, nepřímá nebo virtuální) třídy *classinfo* .

iter(object)

iter(object, sentinel)

Vraťte objekt iterátoru .

len(s)

Vrátí délku (počet položek) objektu.

class list

class list(iterable)

Spíše než být funkcí, list je vlastně proměnlivý sekvenční typ.

locals()

Aktualizujte a vraťte slovník představující aktuální tabulku místních symbolů.

map(function, iterable, *iterables)

Vraťte iterátor, který aplikuje *funkci* na každou *iterovatelnou* položku , přinášející výsledky.

max(iterable, *, key=None)

max(iterable, *, default, key=None)

max(arg1, arg2, *args, key=None)

Vraťte největší položku v iterovatelné nebo největší ze dvou nebo více argumenty.

class memoryview(object)

Vrátí objekt „memory view“ vytvořený z daného argumentu.

min(iterable, *, key=None)

min(iterable, *, default, key=None)

min(arg1, arg2, *args, key=None)

Vraťte nejmenší položku v iterovatelné nebo nejmenší ze dvou nebo více položek argumenty.

next(iterator)

next(iterator, default)

Načtěte další položku z iterátoru zavoláním jeho __next__()metoda.

class object

Vraťte nový objekt bez rysů

oct(x)

Převeďte celé číslo na osmičkový řetězec s předponou „0o“.

otevřít (*soubor* , *režim = 'r '* , *ukládání do vyrovnávací paměti = -1)* , *kódování = žádné* , *chyby = žádné* , *nový = žádné* , *closefd = pravda* , *otvírák = žádné* ¶

Otevřete *soubor* a vraťte odpovídající objekt souboru .

ord(c)

Zadaný řetězec představující jeden znak Unicode vraťte celé číslo představující bod kódu Unicode tohoto znaku.

pow(*base*, *exp*, *mod=None*)

Vraťte *základnu* do power *exp* ; pokud *je mod* přítomen

print(*objects, sep=' ', end='\n', file=None, flush=False)

Vytiskněte *objekty* textového proudu *do souboru*

class property(fget=None, fset=None, fdel=None, doc=None)

Vraťte atribut vlastnosti.

class range(stop)

class range(start, stop, step=1)

Spíše než být funkcí, range je vlastně neměnný typ sekvence

repr(object)

Vrátí řetězec obsahující tisknutelnou reprezentaci objektu.

reversed(seq)

Vraťte zpětný iterátor

round(number, ndigits=None)

Vrátí *číslo* zaokrouhlené s *přesností na n číslic* za desetinnou čárkou

class set

class set(iterable)

Vraťte nový setobjekt, volitelně s prvky převzatými z *iterovatelný*

setattr(object, name, value)

Toto je protějšek getattr(). Argumenty jsou objekt, a řetězec a libovolnou hodnotu.

class slice(stop)

class slice(start, stop, step=1)

Vrátí objekt řezu představující množinu indexů zadanou pomocí range(start, stop, step).

sorted(iterable, /, *, key=None, reverse=False)

Vraťte nový seřazený seznam z položek v *iterable* .

@staticmethod

Transformujte metodu na metodu statickou.

sum(iterable, /, start=0)

Součty *začínají* a položky iterovatelné *zleva* doprava a vrací celkový součet

class str(object='')

class str(object=b'', encoding='utf-8', errors='strict')

Návrat a str verze *objektu*

class super

class super(type, object_or_type=None)

Vrátí objekt proxy, který deleguje volání metody na rodiče nebo sourozence třída *typu* .

class tuple

class tuple(iterable)

Spíše než být funkcí, tuple je vlastně neměnný sekvenční typ

class type(object)

class type(name, bases, dict, **kwds)

S jedním argumentem vrátí typ *objektu* .

vars()

vars(object)

Vraťte __dict__atribut pro modul, třídu, instanci, nebo jakýkoli jiný předmět s a __dict__atribut.

zip(*iterables, strict=False)

Iterujte přes několik iterovatelných paralelně a vytvářejte n-tice s položkou od každého z nich.