Програмування

ТЕМА 4. ЧИСЛОВІ ТИПИ ДАНИХ

Поняття типу даних

Кожен тип даних T визначається такими характеристиками:
- M – множина, з якої набувають значень елементи типу даних – носій типу;
- Ω - множина операцій над елементами типу даних;
- R – множина відношень, які визначені для типу даних;
- I – множина іструкцій, які визначені для елементів типу даних.

Тобто, тип даних – це четвірка:

$T = \{M, \Omega, R, I\}$

Наприклад, для бульового типу даних
- $M = B_2$
- Ω = {or, and, not}
- R = {==, !=, >, <, >=, <=}
- I = {=, print()}

Надалі розглянемо числові типи даних.

Позиційний запис дійсного числа у системі числення за основою b

Визначимо поняття позиційного запису числа в системі числення за довільною натуральною основою $b > 1$.

Позиційним записом числа a за основою *b* назвемо запис наступного вигляду:

$(\ldots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3} \ldots)_b = \ldots + a_3 * b^3 + a_2 * b^2 + a_1 * b^1 + a_0 + + a_{-1} * b^{-1} + a_{-2} * b^{-2} + a_{-3} * b^{-3} + \ldots,$

- де коефіцієнти a_k можуть приймати одне з b значень $0,1,\ldots,b-1$.

Ось приклад позиційного запису за основою 8

$(462.7)_8 = 4*64 + 6*8 + 2 + 7*(1/8) = (306.875)_{10}$.

Функція округлення round

Функцією округлення числа x до p ($p > 1$) значущих цифр у системі числення за основою b назвемо функцію

$round(x, p, b)$,

задану співвідношеннями

- $round(0, p, b) = 0$;
- $round(x, p, b) = sign(x)*b^{r-p}*[b^{p-r}*|x| + 1/2]$,
  - де $b^{r-1} <= |x| < b^r$.

Наприклад:

$round(23.3578, 4, 10) = sign(23.3578)*10^{r-4}*[10^{4-r}*|23.3578| + 1/2] =$

(визначимо r. $10^{r-1} <= |23.3578| < 10^r \Rightarrow r = 2$)

$= 1*10^{2-4}*[10^{4-2}*23.3578 + 0.5] = 10^{-2}*[10^2*23.3578 + 0.5] =$

$= 10^{-2}*[2335.78 + 0.5] = 10^{-2}*[2336.28] = 10^{-2}*2336 = 23.36$

Властивості функції round

a) $round(1, p, b) = 1$.

b) $round(-x, p, b) = -round(x, p, b)$.

c) $round(b*x, p, b) = b*round(x, p, b)$.

Похибка округлення

Абсолютна похибка

$$E = |x - round(x, p, b)|$$

$$E \le \frac{1}{2} b^{r-p}$$

Відносна похибка

$$\delta * |x| = E$$

$$\delta \le \frac{1}{2} b^{1-p}$$

Наближені p-розрядні арифметичні операції над дійсними числами

$u +_p v = round(u+v, p, b);$

$u -_p v = round(u-v, p, b);$

$u *_p v = round(u*v, p, b);$

$u /_p v = round(u/v, p, b);$

Властивості наближених операцій

Наближені p-розрядні арифметичні операції не задовольняють деяким законам звичайної (точної) арифметики.

Залишаються справедливими

a) $u +_p v \equiv v +_p u$, $u *_p v \equiv v *_p u$;

b) $u +_p v = 0 \iff u = -v$;

c) $u *_p v = 0 \iff u = 0$ or $v = 0$;

d) $u /_p u \equiv 1$,

Порушуються

- асоціативність додавання та множення а також дистрибутивність множення відносно додавання.

Приклад порушення асоціативності додавання

$100 +_3 0.4 +_3 0.4$

Обчислимо спочатку

$(100 +_3 0.4) +_3 0.4$

$100 +_3 0.4 = round(100+0.4, 3, 10) = round(100.4, 3, 10) =$

$=sign(100.4)*10^{r-3}*[10^{3-r}*|100.4| + 1/2] =$

$= 1*10^{3-3}*[10^{3-3}*|100.4| + 0.5] = [100.4 + 0.5] = [100.9] = 100 =>$

$(100 +_3 0.4) +_3 0.4 = 100$

Тепер обчислимо

$100 +_3 (0.4 +_3 0.4)$

$0.4 +_3 0.4 = round(0.4+0.4, 3, 10) = round(0.8, 3, 10) =$

$=sign(0.8)*10^{r-3}*[10^{3-r}*|0.8| + 1/2] =$

$= 1*10^{0-3}*[10^{3-0}*|0.8| + 0.5] = 10^{-3}*[10^3*0.8 + 0.5] =$

$=10^{-3}*[800 + 0.5] = 10^{-3}*[800.5] = 10^{-3}*800 = 0.8$

$100 +_3 0.8 = round(100+0.8, 3, 10) = round(100.8, 3, 10) =$

$=sign(100.8)*10^{r-3}*[10^{3-r}*|100.8| + 1/2] =$

$= 1*10^{3-3}*[10^{3-3}*|100.8| + 0.5] = [100.8 + 0.5] = [101.3] = 101$

Пам’ять комп’ютера та змінні

Пам'ять комп’ютера представляє собою послідовність 8-розрядних клітинок - **байтів**.

Кожний розряд - **біт** - може приймати одне з двох значень: 0 або 1.

Для того, щоб дізнатися скільки байтів виділено під деяку змінну або константу, у Python використовують функцію getsizeof().

- Для використання цієї функції у інтерпретаторі треба спочатку набрати

```
from sys import getsizeof
```

а потім, наприклад,

```
getsizeof(5)
```

Дані зберігаються у пам’яті у двійковій системі числення.

Цілий тип даних

<u>носій</u>

- Носієм цілого типу даних є обмежена множина цілих чисел

$Z_n = \{x \in Z\colon |x| \leq n\}$

- Значення n залежить від реалізації цілого типу.
- Наприклад, якщо для збереження цілого числа виділено 4 байти, то з урахуванням виділення 1 біту на знак числа, залишається 31 біт на представлення самого числа.
- Тому в цьому випадку $n = 2^{31}$-1 (насправді мінімальне ціле -2^{31}, а максимальне 2^{31}-1 через те, що 0 включається у додатні числа).

У Python значення *n* попередньо не встановлюється

Можна вважати, що для всіх відомих задач зі скінченним значенням цілого числа, така границя недосяжна

Константи цілого типу позначаються як у звичайній арифметиці, наприклад: 0, 1, -235.

Операції для цілого типу даних

Операція	Опис
x + y	сума x та y
x - y	різниця x та y
x * y	добуток x та y
x / y	частка від ділення x на y (результат – дійсне число)
x // y	ділення націло x на y
x % y	Остача від x // y
-x	x від’ємне
abs(x)	модуль x
int(x)	перетворення x до цілого
float(x)	перетворення x до дійсного
x ** y	x піднесене до степеня y

Відношення та інструкції для цілого типу даних

<u>відношення</u>

Для цілого типу визначені 6 стандартних відношень

==, !=, >, <, >=, <=

<u>інструкції</u>

Визначено присвоєння, введення та виведення

```
x = e, x = int(input(S)), print(x)
```

- Визначено також цикл по діапазону цілих чисел з лічильником цілого типу

```
for i in range(b,c,d):
    P
```

Приклади для цілого типу

Обчислити суму цифр заданого натурального числа та показати всі цифри

Дано натуральне число n. Скласти програму, яка з`ясовує, чи є серед натуральних чисел m <= n такі, для яких можна подати m! у вигляді добутку трьох послідовних натуральних чисел.

Дійсний тип даних

Для визначення носія дійсного типу даних розглянемо відображення *flt*(x,p,b,q)

Число x зображується **наближеним p - розрядним представленням з плаваючою крапкою за основою b з надлишком q** -

flt$(x,p,b,q) = (f, r)$

- де f, r визначаються з співвідношень

$f * b^{r-q} = round(x,p,b);$

$1 <= |f| < b$

- Число f називається мантисою наближення, а r - його порядком. Означення *flt* дозволяє вибрати мантису і порядок однозначно.

Умова $1 <= |f| < b$ називається **умовою нормалізованості**. Наприклад,

flt(0.02241383, 5, 10, 5) = (2.2414, 3)

Дійсний тип даних. Носій

Назвемо **(*p,b,q,d*) - обмеженою множиною дійсних чисел** множину наближених *p* - розрядних представлень з плаваючою крапкою за основою *b* з надлишком *q* та максимальним порядком *d* всіх дійсних чисел $x \in \Delta(q,d)$:

$R_{p,b,q,d} = \{(f,r): (f,r)=flt(x,p,b,q);\ |x| \in \Delta(q,d);\ r \in N_d\}$,

де $N_d = \{n \in N: n \le d\} \cup \{0\}$,

а інтервал представлення Δ (*q, d*) вибирається виходячи з означення *flt* наступним чином

$f * b^{r-q} < b^{r-q+1} \le b^{d-q+1}$

також

$f * b^{r-q} \ge 1 * b^{r-q} \ge b^{-q}$

Дійсний тип даних. Носій.2

Отже,

$\Delta(q,d) = \{0\} \cup \{x \in R$: $x_{min} = b^{-q} \le |x| < b^{d-q+1} = x_{max}\}$.

Причому інтервал

$U = \{x: |x| > x_{max}\}$ називається інтервалом переповнення, а інтервал

$V = \{x: |x| < x_{min}\} \setminus \{0\}$ - інтервалом антипереповнення або зникнення порядку.

Відмітимо, що мінімальний порядок $r_{min} = -q$, а максимальний - $r_{max} = d-q$

Наближені обмежені операції

Визначимо тепер обмежену наближену операцію додавання для дійсних чисел

$$x +. y = \begin{cases} x +_p y,\ якщо\ x + y = 0\ or\ |x + y| \in \Delta(q, d) \\ не\ визначено,\ у\ інших\ випадках \end{cases}$$

- де $x+_p y$ – наближена p – розрядна операція додавання.

Наближені обмежені операції віднімання (-.), множення (*.) та ділення (/.) визначаються аналогічно.

Реалізація носія дійсного типу даних у Python

У Python дійсні числа зберігаються згідно стандарту IEEE 754 у представленні з плаваючою крапкою у двійковій системі числення, тобто $b = 2$.

Для збереження у пам'яті самого числа виділяється 8 байтів (64 біти).

При цьому, 1 біт виділяється під знак числа, 52 біти – під мантису, 11 біт – під порядок.

Числа зберігаються у нормалізованому представленні. Це для двійкової системи числення означає, що старший біт завжди дорівнює 1.

Тому цей старший біт мантиси не зберігають у пам'яті, враховують при виконанні операцій. Таким чином, у пам'яті зберігається частина мантиси f'. Співвідношення між f та f' таке:

$f = (1.f')_2$, тобто, $f = 1+f'$

Реалізація носія дійсного типу даних у Python.2

Оскільки під порядок виділено 11 біт, d = 2047 (2^{11}-1).

Надлишок q = 1023. Враховуючи це, мінімальний порядок числа повинен був бути -1023, а максимальний – 1024.

Але реальний мінімальний порядок r_{min} = -1022, а максимальний – r_{max} = 1023.

- Одиниця з мінімального порядку використовується для збереження 0 (всі нулі у всіх бітах) та маленьких денормалізованих чисел (ненульові біти мантиси), а одиниця максимального порядку – для позначення нескінченності - inf - та випадків коли результат операції не є числом – nan - (наприклад, - корінь з від’ємного числа).

Таким чином, парметри носія дійсного типу даних:

p = 53, b = 2, q = 1023, d = 2047 і маємо $R_{53,2,1023,2047}$

Реалізація носія дійсного типу даних у Python.3

Максимальне число (тут і далі – за модулем)

$x_{max} = 2^{1023} * (2 - 2^{-52}) = 2^{1024} - 2^{971} \approx 2^{1024} \approx 1.8 * 10^{308}$

Мінімальне нормалізоване число

$x_{min} = 1.0 * 2^{-1022} \approx 2.2 * 10^{-308}$.

Мінімальне денормалізоване число

$x'_{min} = 2^{-52} * 2^{-1022} = 2^{-1074} \approx 4.9 * 10^{-324}$.

- Усі числа менші x'_{min} не можна відрізнити від 0.

Оскільки $p = 53$, $b = 2$, кількість вірних десяткових цифр мантиси – 15-16.

Позначення констант дійсного типу

1.2

-356.7569

0.00346

2.3E-3

2.3e-3

Операції для дійсного типу

Основні операції

Операція	Опис
x + y	сума x та y
x - y	різниця x та y
x * y	добуток x та y
x / y	частка від ділення x на y
x // y	ділення націло x на y
x % y	Остача від x // y
-x	x від’ємне
abs(x)	модуль x
int(x)	перетворення x до цілого
float(x)	перетворення x до дійсного
x ** y	x піднесене до степеня y

Відношення та інструкції для дійсного типу

<u>відношення</u>

Для дійсного типу визначені 6 стандартних відношень

==, !=, >, <, >=, <=

<u>інструкції</u>

Визначено присвоєння, введення та виведення

```
x = e, x = float(input(S)), print(x)
```

Додаткові функції для дійсних чисел

Функції доступні, якщо написати `import math`

Функція	Опис
math.pi	Константа π = 3.141592...
math.e	Константа *e* = 2.718281...
math.sqrt(x)	Корінь квадратний з *x*.
math.exp(x)	*e**x*
math.expm1(x)	*e**x* - 1. Для малих x підвищує точність обчислення у порівнянні з exp(x) - 1
math.log(x).	*ln x*
math.log(x, base)	$log_{base}x$
math.log1p(x)	*ln*(1+*x*)
math.log2(x)	$log_2 x$
math.log10(x)	*lg x*
math.cos(x)	*cos x*
math.sin(x)	*sin x*
math.tan(x)	*tg x*

Додаткові функції для дійсних чисел.2

Функція	Опис
math.acos(x)	*arccos x*
math.asin(x)	*arcsin x*
math.atan(x)	*arctg x*
math.atan2(y, x)	math.atan(y / x)
math.hypot(x, y)	math.sqrt(x*x + y*y)
math.degrees(x)	Перетворює *x* з радіан на градуси
math.radians(x)	Перетворює *x* з градусів на радіани
math.cosh(x)	*cosh x*
math.sinh(x)	*sinh x*
math.tanh(x)	*tgh x*
math.factorial(x)	*x*! *x* повинен бути цілим та невід’ємним

Додаткові функції для дійсних чисел.3

Функція	Опис
math.isfinite(x)	Істина (True), якщо *x* скінченне дійсне число
math.isinf(x)	Істина (True), якщо *x* нескінченне (inf)
math.isnan(x)	Істина (True), якщо *x* не є дійсним числом (nan)
math.trunc(x)	Дійсне число, яке є результатом відкидання дробової частини *x*
math.ceil(x)	Найменше ціле число, яке більше або рівне *x*
math.floor(x)	Найбільше ціле число, яке менше або рівне *x*

Приклад для дійсних чисел

Наближене обчислення синуса

Комплексний тип даних

<u>носій</u>

Носієм комплексного типу даних є декартів добуток множин

$C = R_{p,b,q,d} \times R_{p,b,q,d}$,

- де $R_{p,b,q,d}$ – носій дійсного типу

Створити комплексне число можна, ввівши *a*+*b*j. Наприклад, z = 2 + 3j.

Таким же чином позначають і константи комплексного типу, наприклад, 1.2+0.44j

Операції для комплексного типу

Основні операції

Операція	Опис
x + y	сума x та y
x - y	різниця x та y
x * y	добуток x та y
x / y	частка від ділення x на y
-x	x від’ємне
abs(x)	модуль x
complex(re, im)	Створення комплексного числа з пари дійсних чисел re та im
x.real	дійсна частина x
x.imag	уявна частина x
x. conjugate()	комплексно-спряжене число до x
x ** y	x піднесене до степеня y

Відношення та інструкції для комплексного типу

<u>відношення</u>

Для комплексного типу визначені відношення

==, !=

<u>інструкції</u>

Визначено присвоєння, введення та виведення

```
x = e, x = complex(input(S)), print(x),
```

Додаткові функції для комплексних чисел

Функції доступні, якщо написати `import cmath`

Функція	Опис
cmath.pi	Константа π = 3.141592...
cmath.e	Константа e = 2.718281...
cmath.sqrt(x)	Корінь квадратний з x.
cmath.exp(x)	e**x
cmath.log(x).	ln x
cmath.log(x, base).	$\log_{base} x$
cmath.log10(x)	lg x
cmath.cos(x)	cos x
cmath.sin(x)	sin x
cmath.tan(x)	tg x
cmath.acos(x)	arccos x
cmath.asin(x)	arcsin x
cmath.atan(x)	arctg x

Додаткові функції для комплексних чисел.2

Функція	Опис
cmath.cosh(x)	cosh x
cmath.sinh(x)	sinh x
cmath.tanh(x)	tgh x
cmath.isfinite(x)	Істина (True), якщо обидві частини x скінченні дійсні числа
cmath.isinf(x)	Істина (True), якщо хоча б одна з частин x нескінченне (inf)
cmath.isnan(x)	Істина (True), якщо хоча б одна з частин x не є дійсним числом (nan)
cmath.phase(x)	Аргумент x
cmath.polar(x)	Представлення x у полярних координатах
cmath.rect(r,phi)	Комплексне число з полярними координатами r, phi

Приклад для комплексних чисел

Знайти усі розв’язки рівняння

$ax^2 + bx + c = 0$

з дійсними коефіцієнтами

Резюме

Ми розглянули:

1. Поняття типу даних. Складові частини типу даних.
2. Позиційний запис дійсного числа.
3. Функцію округлення, похибку округлення.
4. Наближені операції та їх властивості
5. Цілий тип даних.
6. Дійсний тип даних.
7. Комплексний тип даних

Де прочитати

1. Обвінцев О.В. Інформатика та програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. A Byte of Python (Russian) Версия 2.01 Swaroop C H (Translated by Vladimir Smolyar), http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf
3. Бублик В.В., Личман В.В., Обвінцев О.В.. Інформатика та програмування. Електронний конспект лекцій, 2003 р.,
4. Марк Лутц, Изучаем Python, 4-е издание, 2010, Символ-Плюс
5. Python 3.4.3 documentation
6. http://deeplearning.net/software/theano/tutorial/python-memory-management.html
7. http://www.laurentluce.com/posts/python-integer-objects-implementation/
8. https://docs.python.org/2/tutorial/floatingpoint.html
9. http://www.johndcook.com/blog/2009/04/06/anatomy-of-a-floating-point-number/