

Змінні, вирази та інструкції

Розділ 2

Python для всіх
www.py4e.com



Константи

- Сталі значення як-от цілі числа, літери, рядки називаються «константами», бо їхнє значення не змінюється
- Числові константи зазвичай такі, як ви й прогнозували
- Для рядкових констант використовують одинарні (') або подвійні лапки (")

```
>>> print(123)
```

```
123
```

```
>>> print(98.6)
```

```
98.6
```

```
>>> print('Hello world')
```

```
Hello world
```

Зарезервовані (ключові) слова

Ви не можете застосувати **зарезервовані слова** для позначення назв змінних / ідентифікаторів

```
False  class  return  is  finally  
None   if     for    lambda  continue  
True   def    from   while  nonlocal  
and    del    global not    with  
as     elif   try    or     yield  
assert else  import pass  
break  except in     raise
```

Змінні

- **Змінна** – це іменоване місце в пам'яті, де програміст зберігає дані, щоб потім отримувати їх, лише написавши «ім'я» **змінної**
- Програміст самостійно обирає назви **змінних**
- Значення **змінних** можна змінювати за допомогою інструкцій

x = 12.2

y = 14

x

12.2

y

14

Змінні

- **Змінна** – це іменоване місце в пам'яті, де програміст зберігає дані, щоби потім отримувати їх, лише написавши «ім'я» **змінної**
- Програміст самостійно обирає назви **змінних**
- Значення **змінних** можна змінювати за допомогою інструкцій

x = 12.2

y = 14

x = 100

x

~~12.2~~

y

14

Правила найменування змінних у Python

- Необхідно починати з літери або підкреслення _
- Містять літери, цифри, підкреслення
- Чутливі до регістру

Придатні: spam eggs spam23 _speed

Непридатні: 23spam #sign var.12

Різні: spam Spam SPAM

Мнемонічні назви змінних

- Оскільки ми, програмісти, можемо обирати, як називати змінні, відома така «найкраща практика»
- Ми називаємо змінні так, щоб запам'ятати, які саме дані зберігатимемо («**мнемоніка**» = «допомога для пам'яті»)
- Інтуїтивні назви можуть збивати початківців з пантелику оскільки вдало названі змінні часто «звучать» так милозвучно, що схожі на ключові слова

<http://en.wikipedia.org/wiki/Mnemonic>

```
x1q3z9ocd = 35.0  
x1q3z9afd = 12.50  
x1q3p9afd = x1q3z9ocd * x1q3z9afd  
print(x1q3p9afd)
```

Що робить цей
фрагмент коду?


```
x1q3z9ocd = 35.0  
x1q3z9afd = 12.50  
x1q3p9afd = x1q3z9ocd * x1q3z9afd  
print(x1q3p9afd)
```

```
a = 35.0  
b = 12.50  
c = a * b  
print(c)
```

Що роблять ці
фрагменти коду?

```
x1q3z9ocd = 35.0  
x1q3z9afd = 12.50  
x1q3p9afd = x1q3z9ocd * x1q3z9afd  
print(x1q3p9afd)
```

```
a = 35.0  
b = 12.50  
c = a * b  
print(c)
```

Що роблять ці
фрагменти коду?

```
hours = 35.0  
rate = 12.50  
pay = hours * rate  
print(pay)
```

Речення чи рядки коду

x = 2



Інструкція присвоювання

x = x + 2



Присвоєння з виразом

print(x)



Інструкція виводу на екран

Змінна

Оператор

Константа

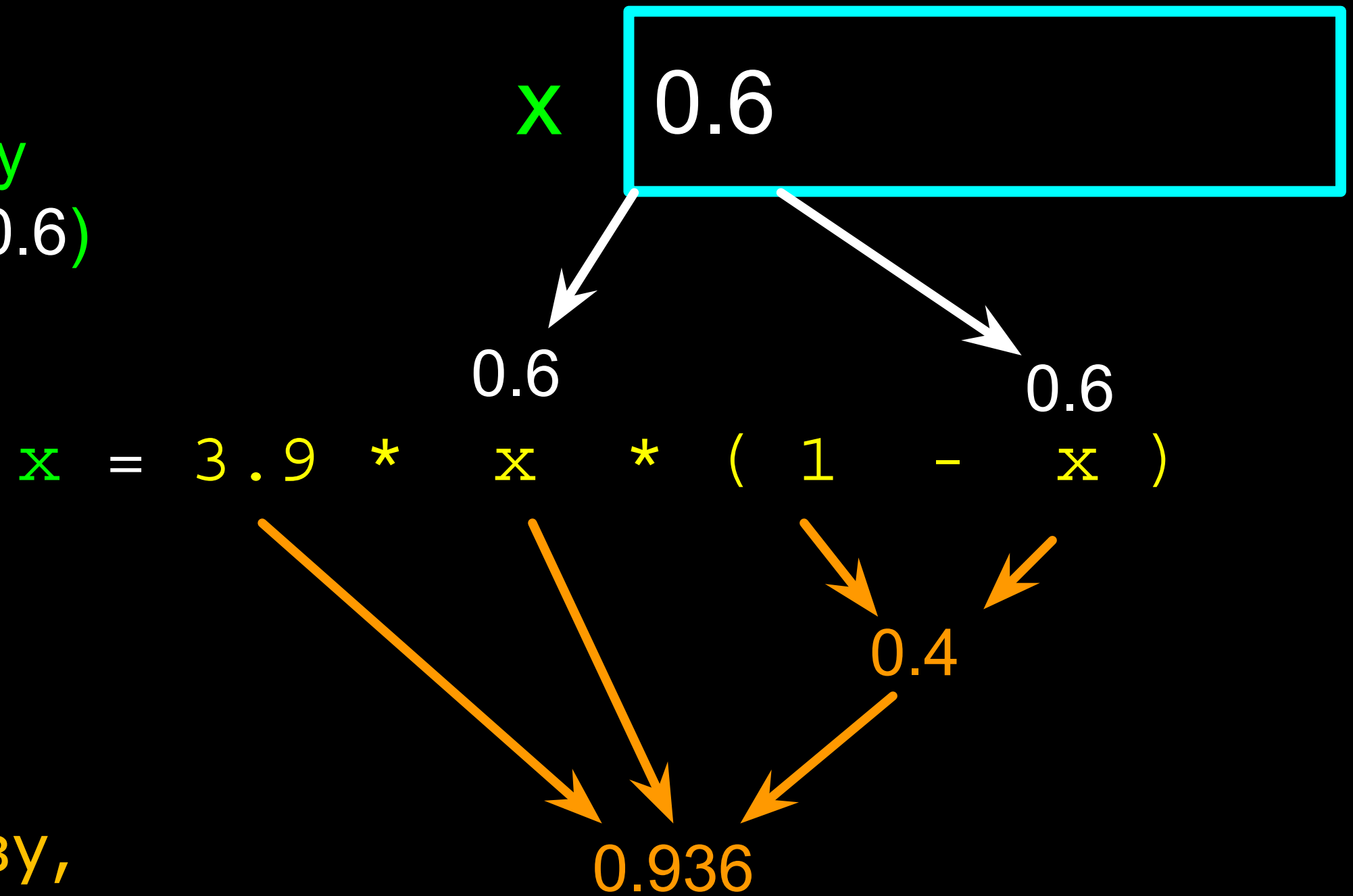
Функція

Інструкції присвоювання

- Ми присвоюємо значення змінній за допомогою інструкції присвоювання (=)
- Інструкція присвоювання складається з виразу **у правій частині** та **змінна** для зберігання результату у лівій частині

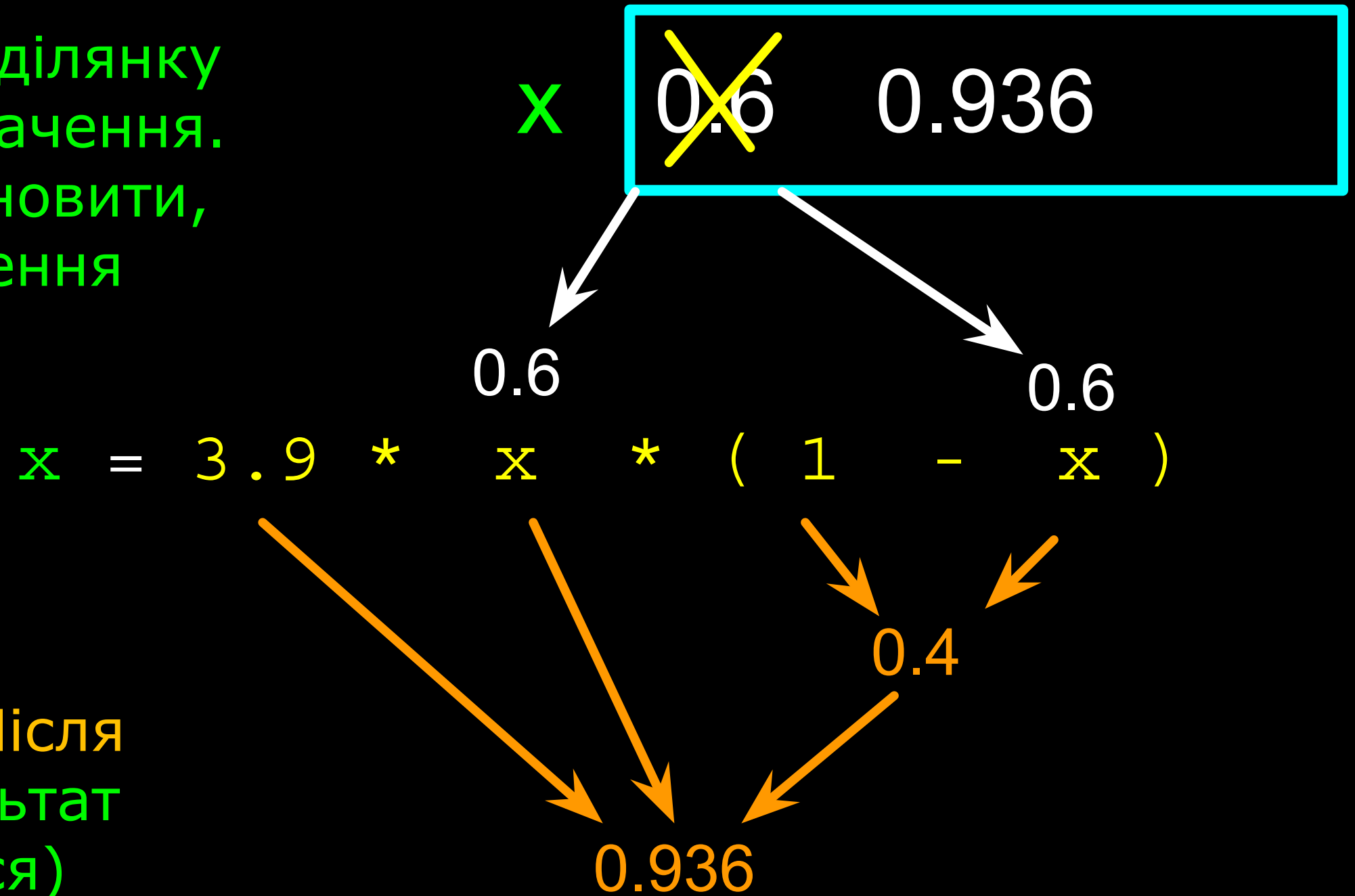
x = 3.9 * **x** * (1 - **x**)

Змінна – це посилання на ділянку пам'яті, призначену для зберігання значення (0.6)



Праворуч бачимо вираз.
Після обчислення виразу,
результат поміщається
(присвоюється) у змінну x .

Змінна – це посилання на ділянку пам'яті, де зберігається значення. Значення змінної можна оновити, тобто замінити старе значення (0.6) на нове (0.936)



Праворуч бачимо вираз. Після обчислення виразу, результат поміщається (присвоюється) змінній у лівій частині (як-от x).

Вирази...

Числові вирази

- Через брак математичних знаків на комп'ютерних клавіатурах, ми почали використовувати інші клавіші для класичних математичних операцій.
- Зірочка – це множення.
- Піднесення до степеня також відрізняється від математичного.

Оператор	Операція
+	Додавання
-	Віднімання
*	Множення
/	Ділення
**	Степень
%	Остача

Числові вирази

```
>>> xx = 2
>>> xx = xx + 2
>>> print(xx)
4
>>> yy = 440 * 12
>>> print(yy)
5280
>>> zz = yy / 1000
>>> print(zz)
5.28
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print(kk)
3
>>> print(4 ** 3)
64
```

5 $\overline{) 23}$
20

3

4 R 3

Оператор	Операція
+	Додавання
-	Віднімання
*	Множення
/	Ділення
**	Степень
%	Остача

Порядок обчислення

- Коли ми використовуємо разом оператори – Python має знати, що робити спочатку, а що потім
- Це називається «порядком обчислення»
- Який оператор «має перевагу» над іншими?

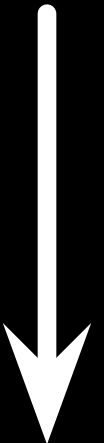
`x = 1 + 2 * 3 - 4 / 5 ** 6`

Правила пріоритетності операторів

Від найвищого пріоритету до найнижчого пріоритету:

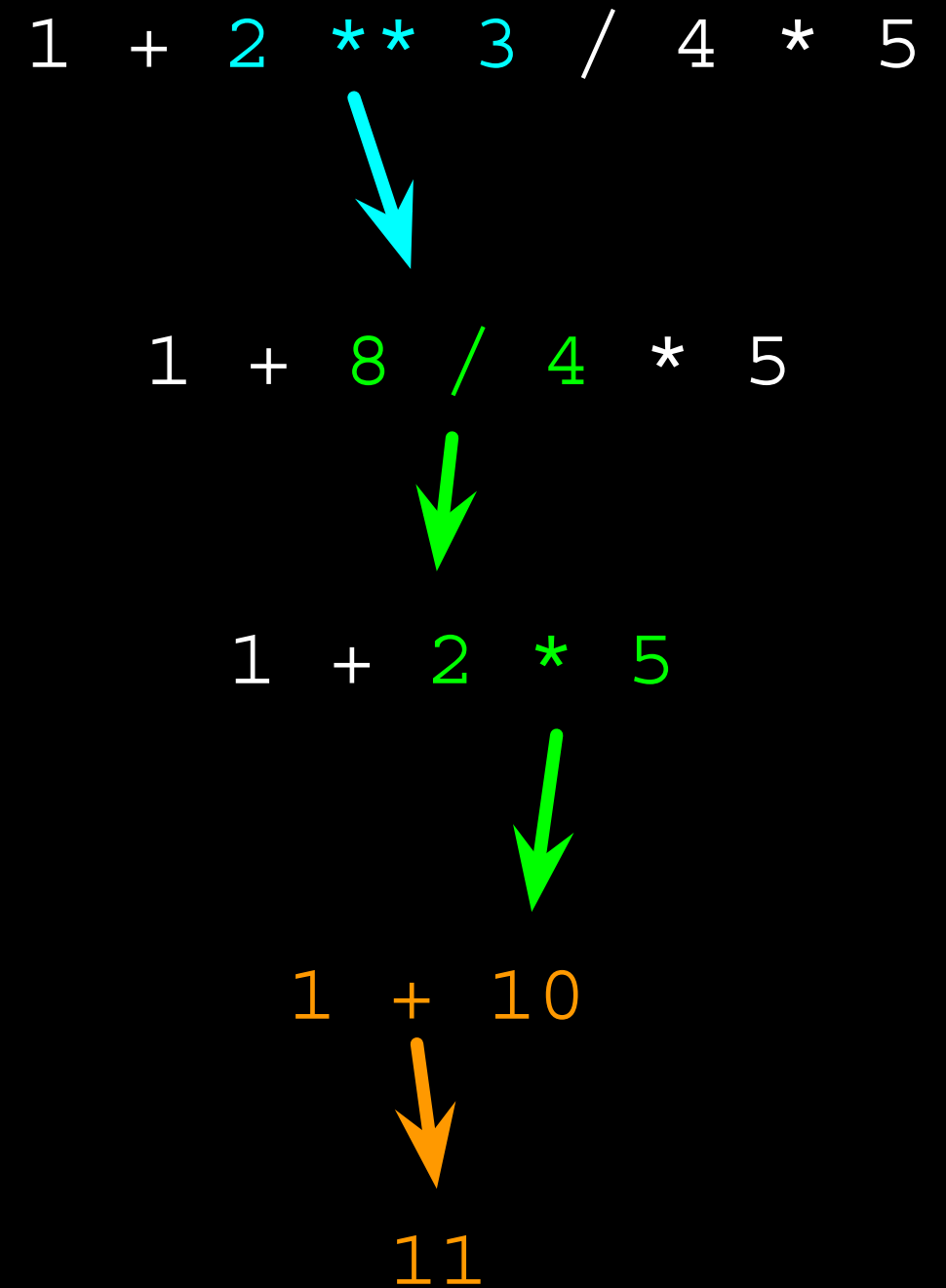
- Дужки завжди в пріоритеті
- Піднесення до степеня
- Множення, ділення, остача
- Додавання або віднімання
- Зліва направо

Дужки
Степень
Множення
Додавання
Зліва направо



```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11.0
>>>
```

Дужки
Степень
Множення
Додавання
Зліва направо



Пріоритетність операторів

Дужки
Степень
Множення
Додавання
Зліва направо



- Запам'ятайте правила пріоритетності
- При написанні коду використовуйте дужки
- Спростуйте математичні вирази у кодї, щоб їх було легко зрозуміти
- Розбивайте довгі переліки математичних операцій на кілька, щоб краще розуміти порядок виконання

Що означає «тип»?

- У Python змінні, рядки і константи мають «тип»
- Python знає різницю між цілим числом та рядком
- Наприклад, «+» означає «додавання», якщо стоїть поряд з числом, або може використовуватися для з'єднання рядків

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

конкатинувати = об'єднувати

Тип важливий

- Python знає **типи** всього
- Деякі операції є забороненими
- Ви не можете «додати 1» у рядок
- Ми можемо дізнатися у Python «що це за тип» за допомогою функції **type()**

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):
File "<stdin>", line 1, in
<module>TypeError: Can't convert
'int' object to str implicitly
>>> type(eee)
<class'str'>
>>> type('hello')
<class'str'>
>>> type(1)
<class'int'>
>>>
```

Деякі типи чисел

- Числа поділяють на два основних типи:
 - **цілі (integers)**, серед яких числа типу -14, -2, 0, 1, 100, 401233
 - **числа з плаваючою крапкою (float)**, що мають десяткове значення після цілого: -2.5, 0.0, 98.6, 14.0
- Відомі й інші типи чисел – різновиди чисел з рухомою крапкою та цілих

```
>>> xx = 1
>>> type (xx)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class 'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
>>>
```


Конвертування типів

- Коли у виразі є ціле число (integer) і число з плаваючою крапкою (float), результат виразу **автоматично** буде числом з плаваючою крапкою (float)
- Ви можете контролювати це за допомогою вбудованих функцій `int()` і `float()`

```
>>> print(float(99) + 100)
199.0
>>> i = 42
>>> type(i)
<class 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class 'float'>
>>>
```

Ділення цілого числа

При діленні отримуєте результат з плаваючою крапкою (float)

Ділення цілих чисел у Python 2 відбувається інакше

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

Перетворення рядків

- Ви також можете використовувати `int()` і `float()`, щоби перетворити рядок на число
- Вам видасть **помилку**, якщо у рядку немає числових символів

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object
to str implicitly
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
with base 10: 'x'
```

Введення даних користувачем

- Ви можете навчити Python зупинятися та читати дані від користувача за допомогою функції `input()`
- Функції `input()` повертає рядок

```
nam = input('Who are you? ')\nprint('Welcome', nam)
```

Who are you? **Chuck**
Welcome Chuck

Конвертування вводу користувача



- Якщо ми хочемо прочитати число від користувача, нам треба конвертувати його з рядка в число за допомогою функції конвертування
- Пізніше ми розглянемо, що робити із невдалими вхідними даними

```
inp = input('Europe floor? ')\nusf = int(inp) + 1\nprint('US floor', usf)
```

Europe floor? 0
US floor 1

Коментарі у Python

- Усе після **#** ігнорується Python
- Навіщо коментувати?
 - Пояснити, що відбуватиметься в зазначеному сегменті коду
 - Зазначити автора коду або іншу допоміжну інформацію
 - Вимкнути певні рядки коду (можливо, тимчасово)

```
# Отримати ім'я файлу та відкрити його
name = input('Enter file:')
handle = open(name, 'r')

# Рахувати частоту слів
counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

# Знайти найуживаніше слово
bigcount = None
bigword = None
for word, count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

# Готово
print(bigword, bigcount)
```

Підсумки

- Тип
- Ключові (зарезервовані) слова
- Назви (мнемонічні)
- Оператори
- Пріоритет операторів
- Ділення цілих чисел
- Конвертація типів
- Ввід користувача
- Коментарі (#)

Вправа

Напишіть програму, яка обчислить
номінальну заробітну плату користувача
відповідно до кількості робочих годин і
ставки на годину

Години: 35

Ставка на годину: 2.75

Оплата: 96.25



Права власності / Застереження



Авторські права на ці слайди з 2010 року належать Чарльзу Северенсу (www.dr-chuck.com) зі Школи інформації Мічиганського університету та застережені ліцензією Creative Commons Attribution 4.0. Будь ласка, збережіть цей фінальний слайд у всіх копіях документа, щоб відповідати вимогам ліцензії щодо посилань на джерела. При повторній публікації матеріалів, якщо щось зміните, додайте ім'я та організацію до переліку співавторів нижче.

Першоджерело: Чарльз Северенс, Школа інформації Мічиганського університету

Переклад: Платформа Prometheus