

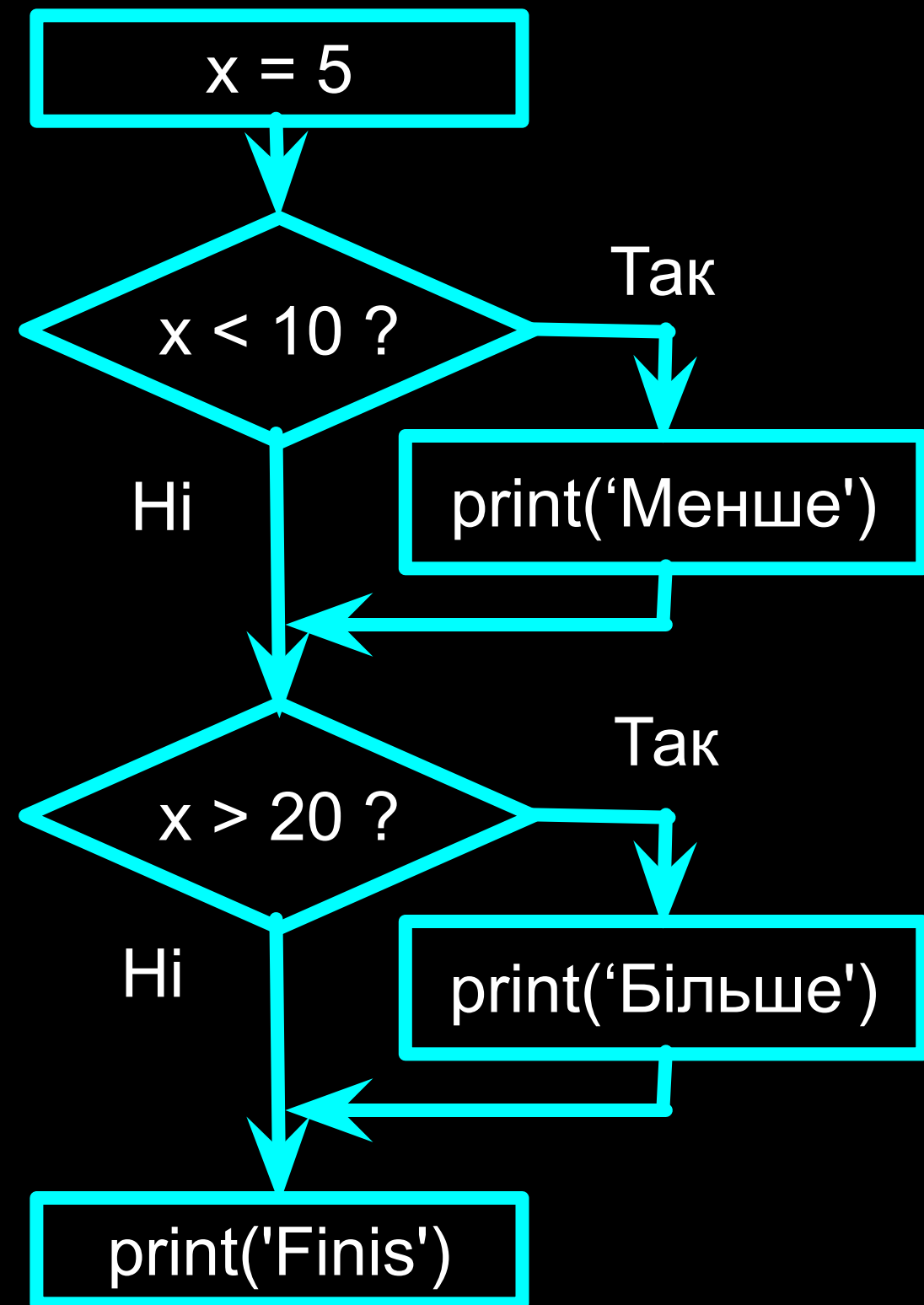
УМОВНЕ ВИКОНАННЯ

Розділ 3

Пайтон для всіх
www.py4e.com



УМОВНЕ ВИКОНАННЯ (крок за кроком)



Програма:

```
x = 5
if x < 10:
    print('Менше')
if x > 20:
    print('Більше')
print('Finis')
```

Вивід:

Менше
Finis

Оператори порівняння

- **Логічні вирази** ставлять запитання і видають результат «Так» або «Ні», який ми використовуємо для керування програмою
- **Логічні вирази** з використанням **операторів порівняння** повертають значення Істина/Хибність (True/False) або Так/Ні
- Оператори порівняння переглядають змінні, але не змінюють їх значень

Python	Значення
<	Менше ніж
<=	Менше або дорівнює
==	Дорівнює
>=	Більше або дорівнює
>	Більше ніж
!=	Не дорівнює

Зауважте: «=» використовується для присвоєння.

http://en.wikipedia.org/wiki/George_Boole

Оператори порівняння

<pre>x = 5</pre>	
<pre>if x == 5 : print('Дорівнює 5')</pre>	Дорівнює 5
<pre>if x > 4 : print('Більше ніж 4')</pre>	Більше ніж 4
<pre>if x >= 5 : print('Більше або дорівнює 5')</pre>	Більше або дорівнює 5
<pre>if x < 6 : print('Менше ніж 6')</pre>	Менше ніж 6
<pre>if x <= 5 : print('Менше або дорівнює 5')</pre>	Менше або дорівнює 6
<pre>if x != 6 : print('Не дорівнює 6')</pre>	Не дорівнює 6

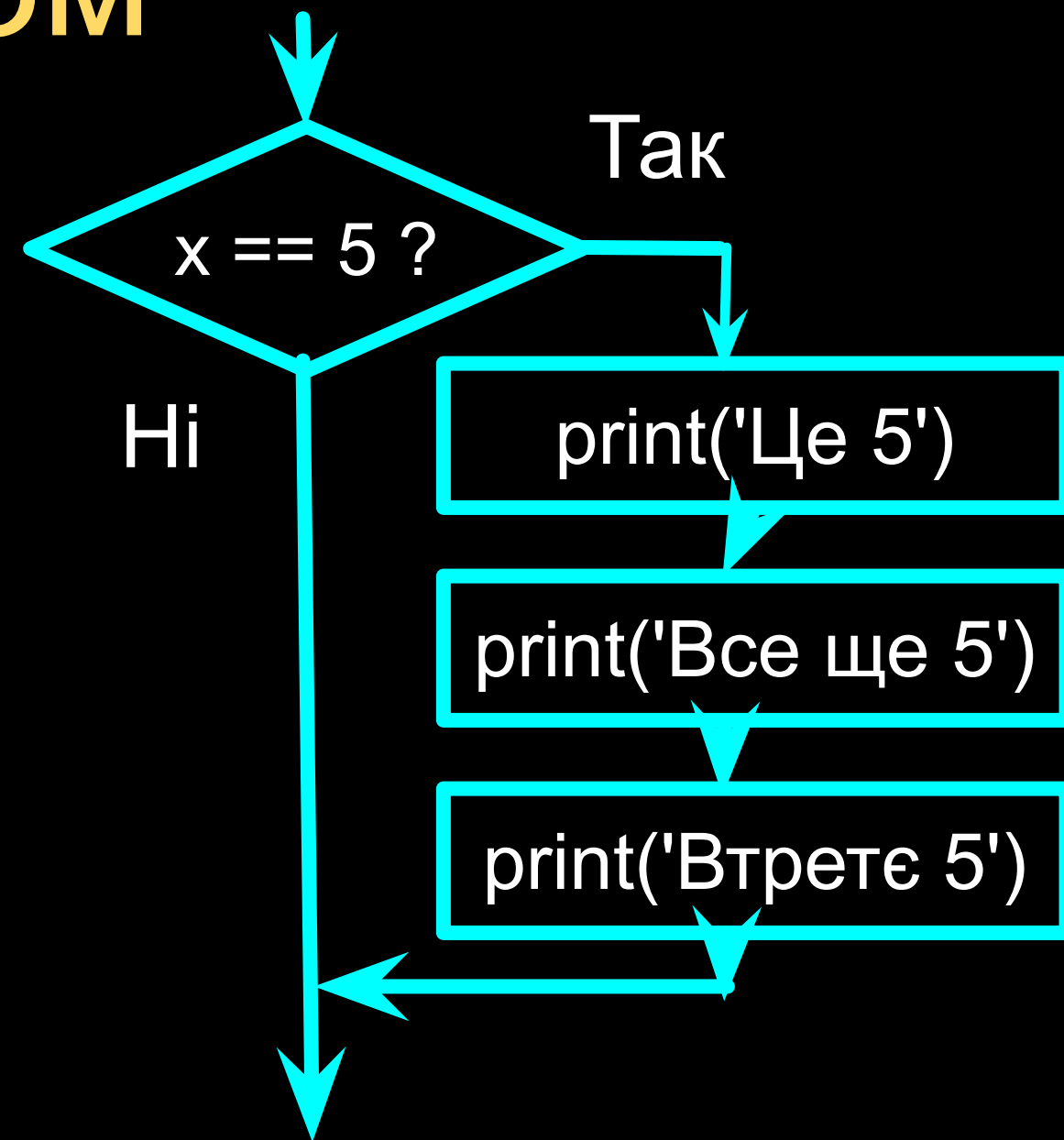
УМОВИ З ОДНИМ ШЛЯХОМ

```
x = 5
print('Before 5')
if x == 5 :
    print('Це 5')
    print('Все ще 5')
    print('Втретє 5')
print('Після 5')
print('Перед 6')
if x == 6 :
    print('Is 6')
    print('Is Still 6')
    print('Third 6')
print('Після 6')
```

До 5

Це 5
Все ще 5
Втретє 5
Після 5
Перед 6

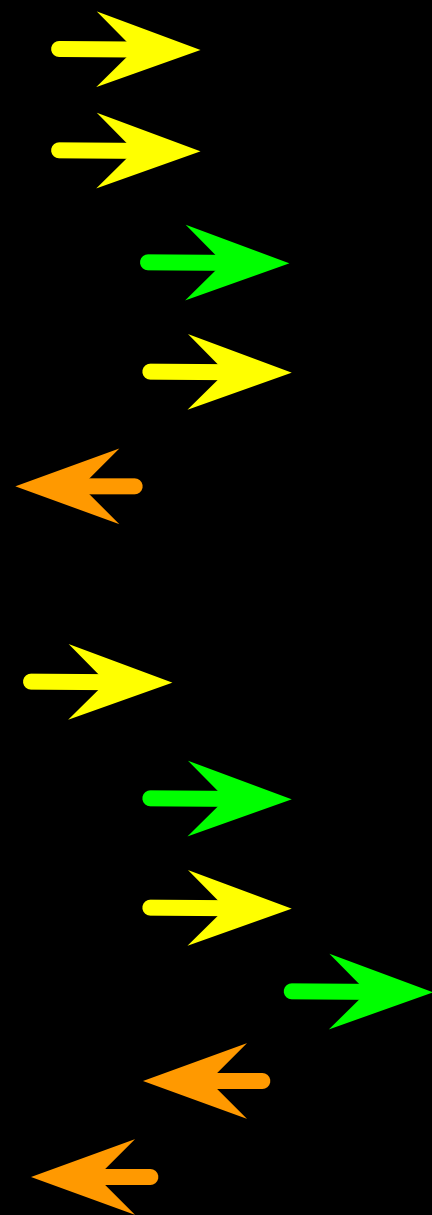
Після 6



Відступи

- **Збільшити відступ** – відступ після оператора **if** чи **for** (після :)
- **Зберегти відступ**, аби позначити що ви **досі в блоці інструкцій** (рядки керуються операторами **if / for**)
- **Зменшити відступ** до рівня оператора **if** або **for**, щоб вказати на кінець блока
- **Порожні рядки** ігноруються – вони не впливають на **відступи**
- **Коментарі** до рядка самі по собі ігноруються, незалежно до **відступів**

додати / залишити після if або for
прибрати відступ для позначення кінця блока



```
x = 5
if x > 2 :
    print('Bigger than 2')
    print('Still bigger')
print('Done with 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Bigger than 2')
    print('Done with i', i)
print('All Done')
```

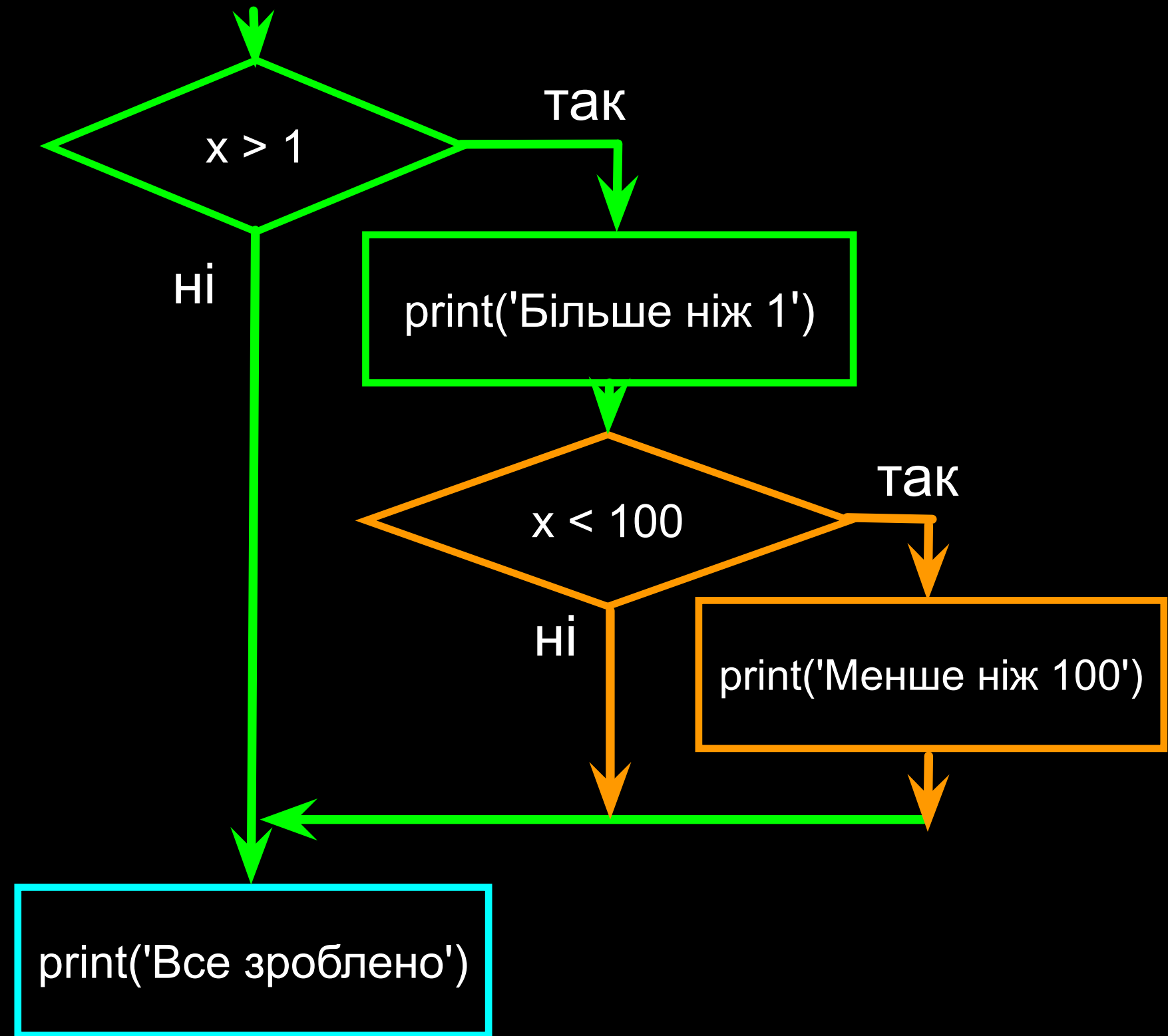
Подумайте про початок / кінець блока

```
x = 5
if x > 2 :
    print('Bigger than 2')
    print('Still bigger')
print('Done with 2')
```

```
for i in range(5) :
    print(i)
    if i > 2 :
        print('Bigger than 2')
    print('Done with i', i)
print('All Done')
```

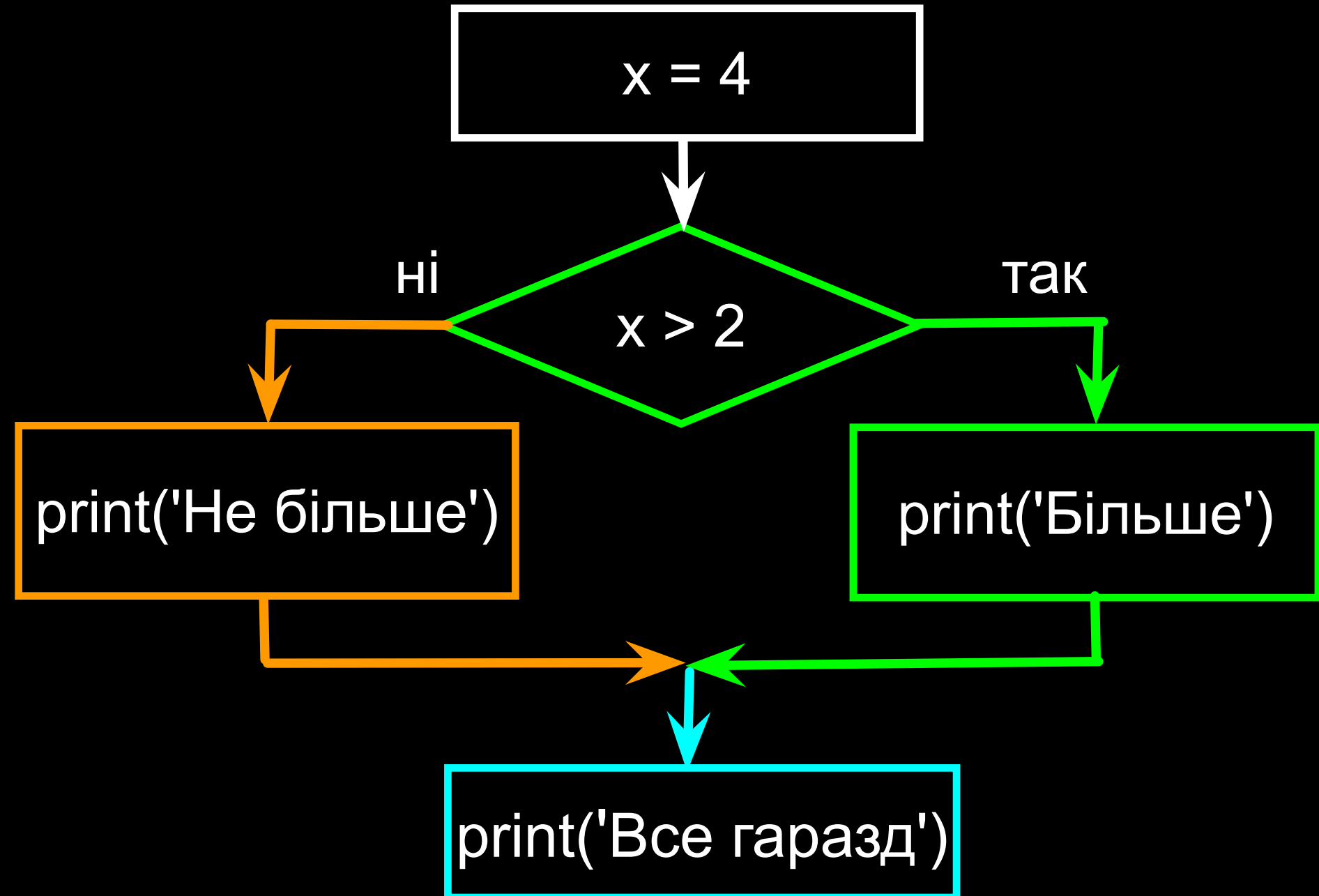

Вкладені умови

```
x = 42
if x > 1 :
    print('Більше ніж 1')
    if x < 100 :
        print('Менше ніж 100')
print('Все зроблено')
```



УМОВИ З ДВОМА ШЛЯХАМИ

- Іноді ми хочемо зробити щось одне, якщо твердження істинне, і щось інше, якщо твердження хибне
- Це як розвилка на дорозі – ми маємо обрати **один або інший** шлях, але не обидва

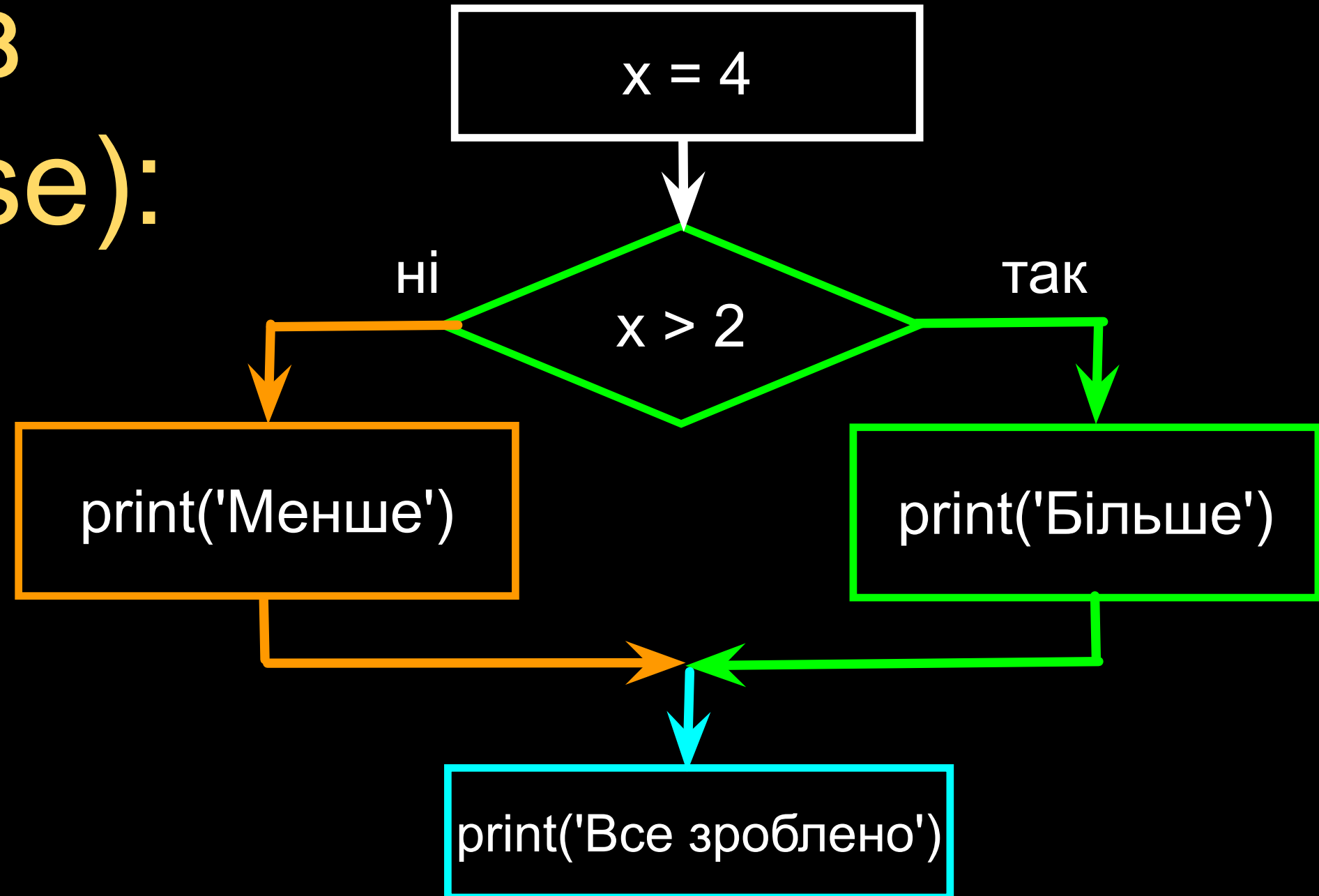


УМОВИ З ДВОМА ШЛЯХАМИ З «інакше» (else):

x = 4

```
if x > 2 :  
    print('Більше')  
else :  
    print('Менше')
```

```
print('Все зроблено')
```

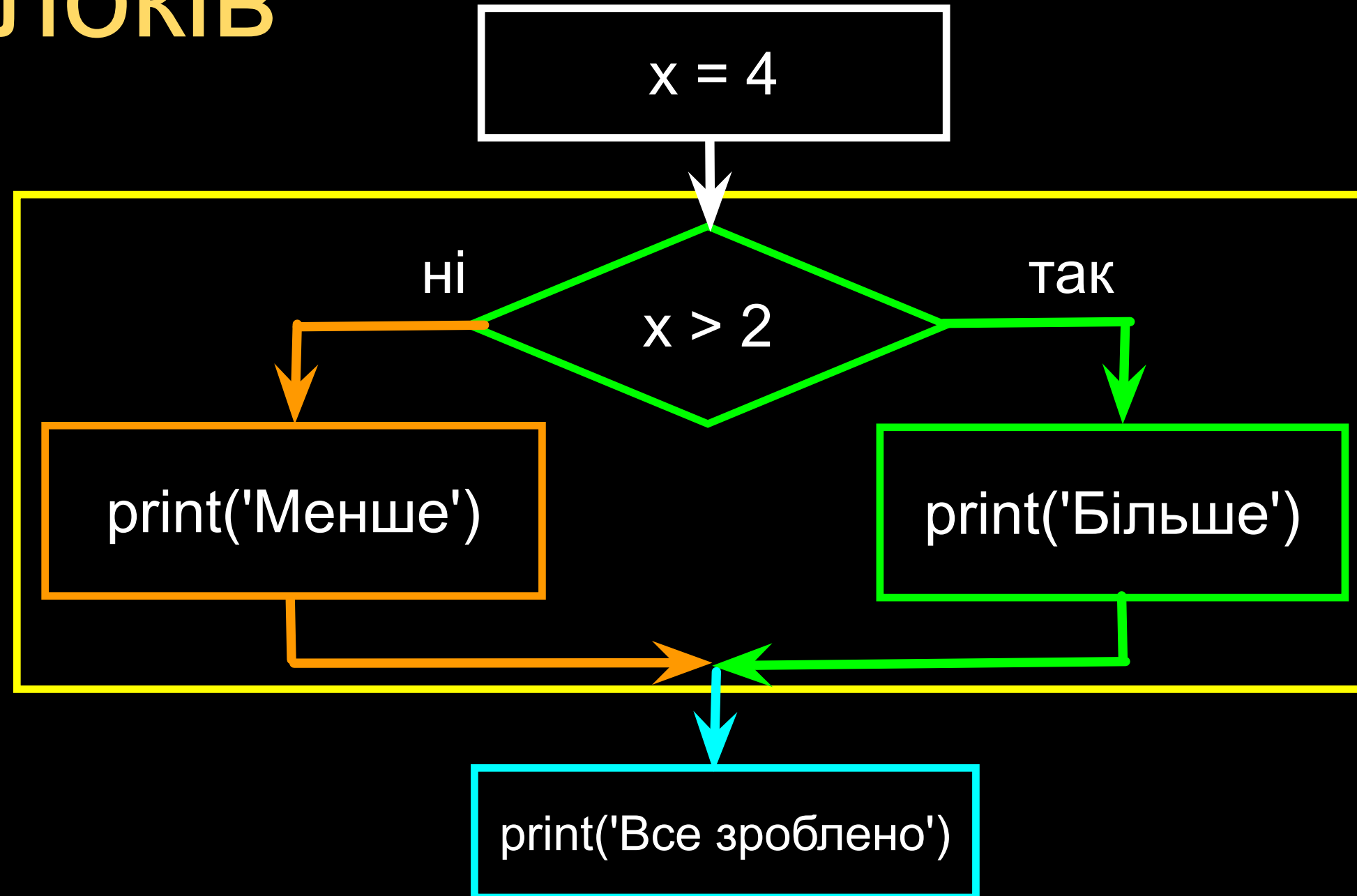


Візуалізація блоків

x = 4

```
if x > 2 :  
    print('Більше')  
else :  
    print('Менше')
```

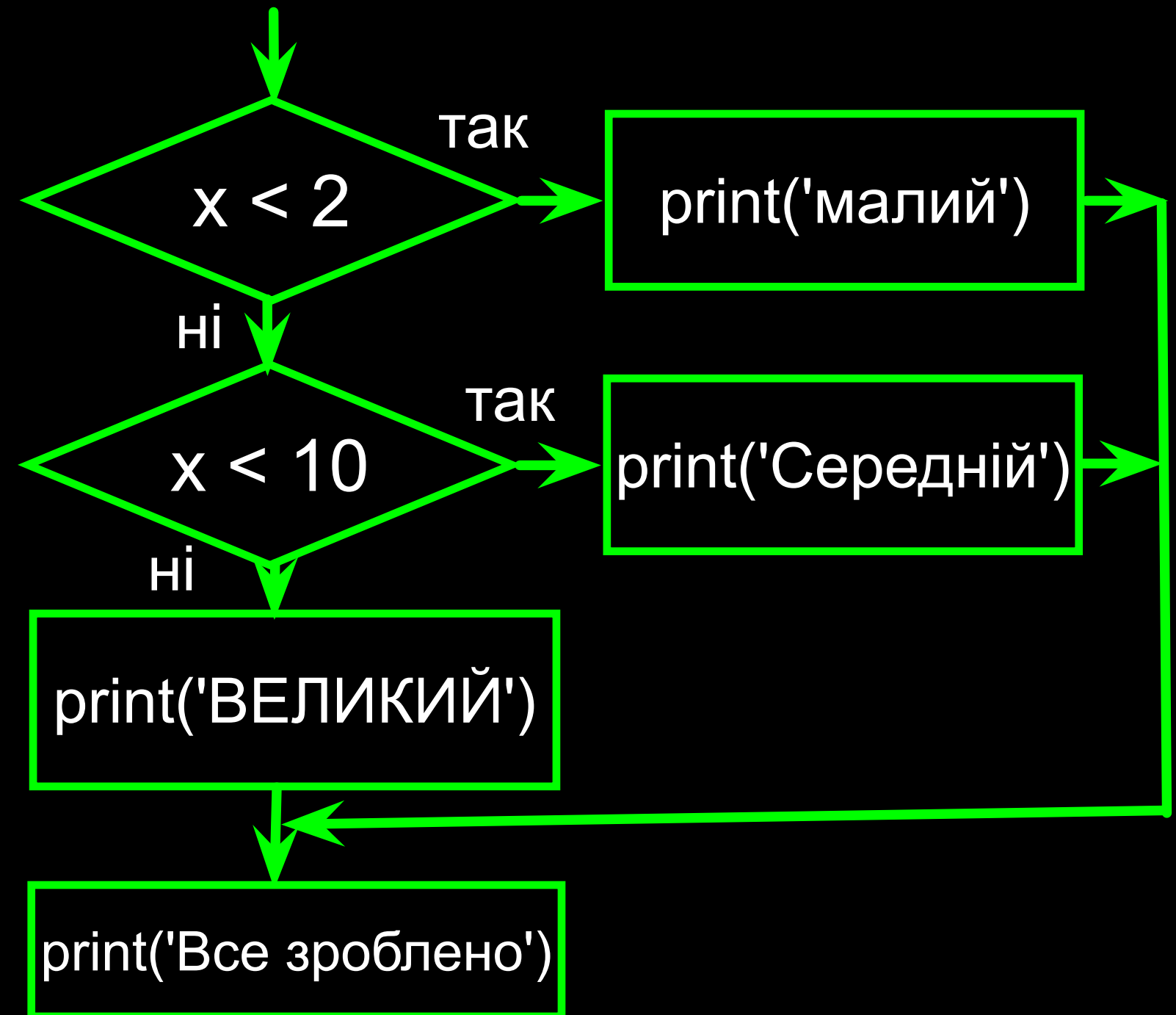
```
print('Все зроблено')
```



Більше умовних конструкцій...

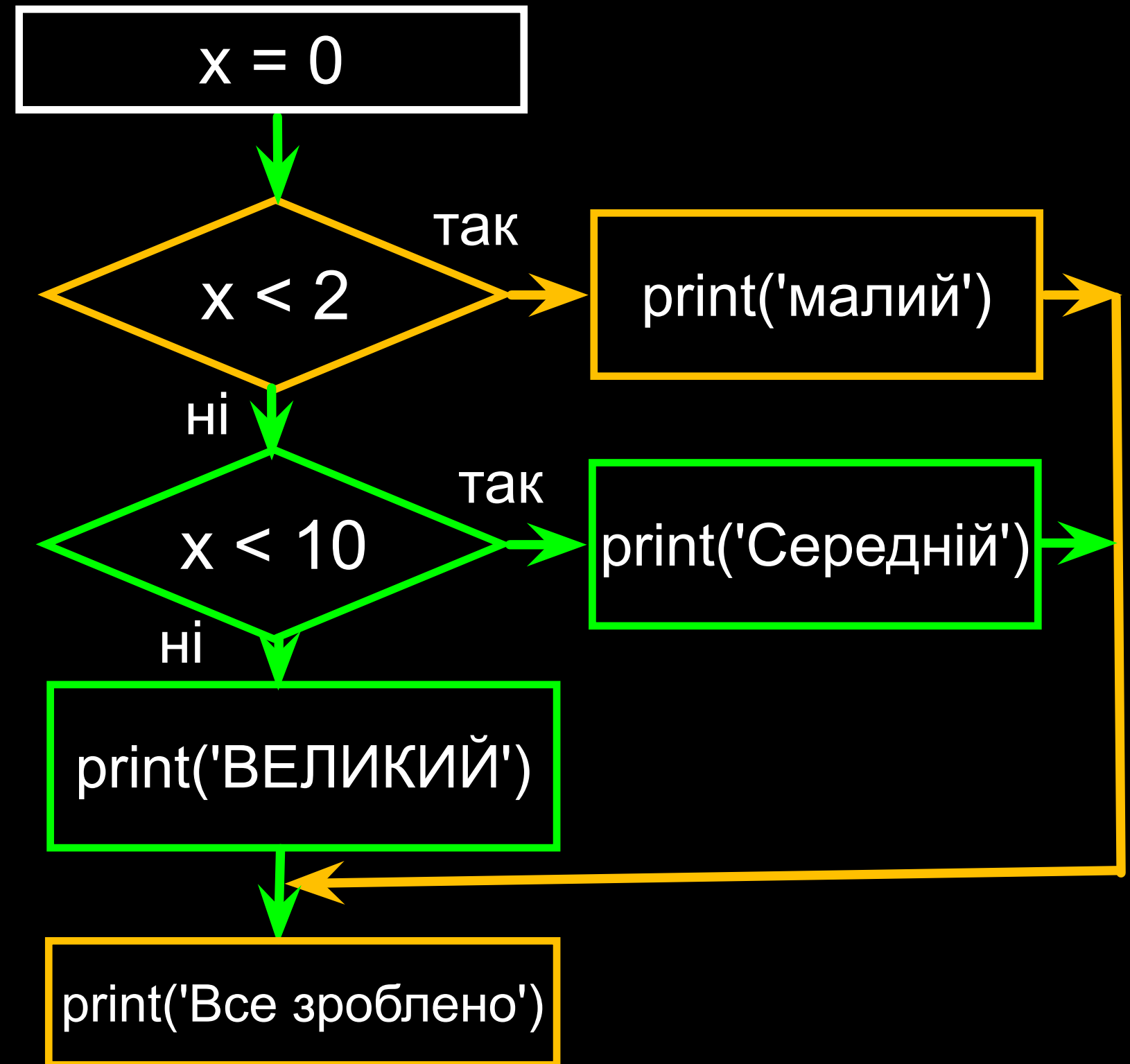
Множинне розгалуження

```
if x < 2 :  
    print('малий')  
elif x < 10 :  
    print('Середній')  
else :  
    print('ВЕЛИКИЙ')  
print('Все зроблено')
```



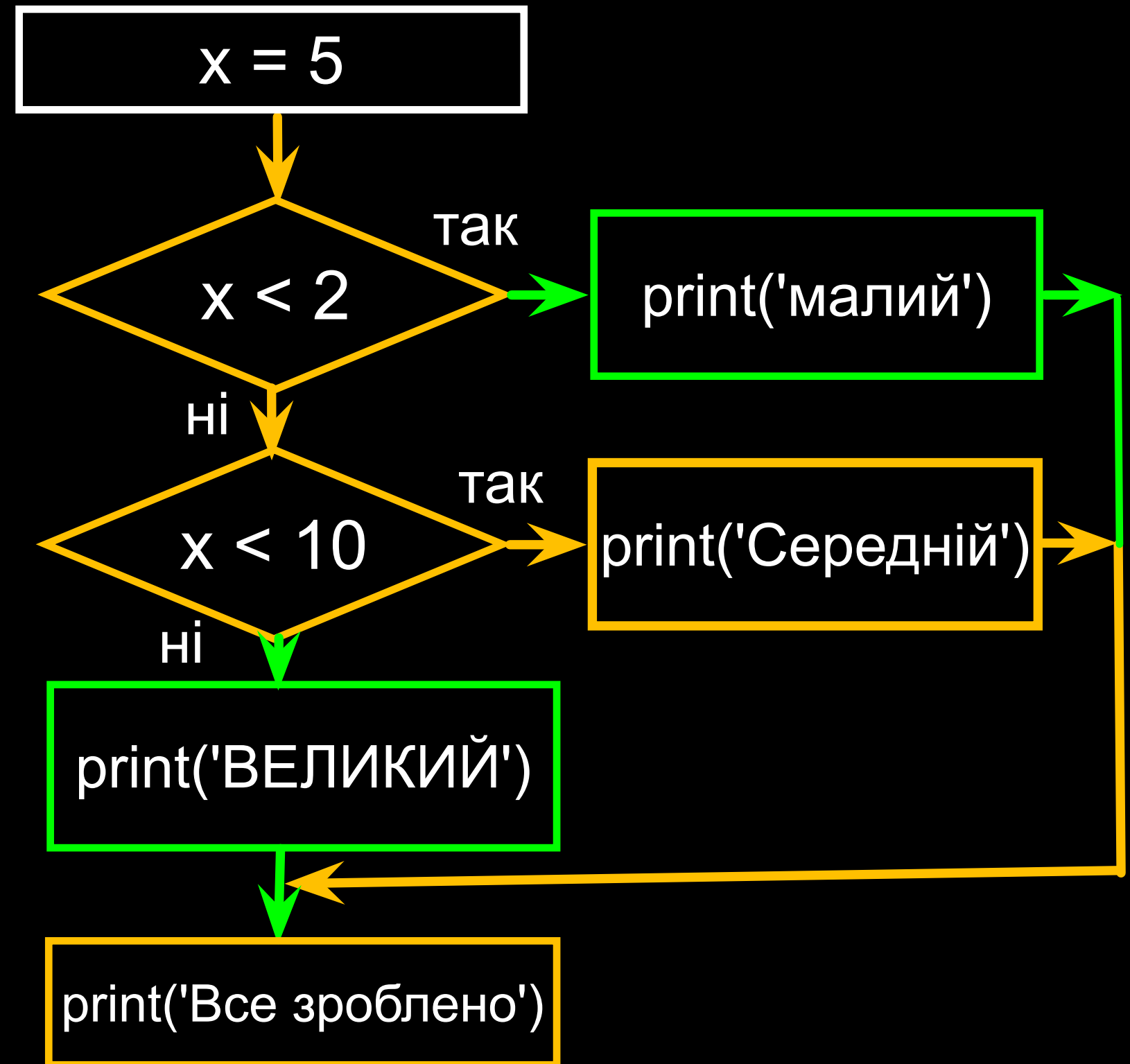
Множинне розгалуження

```
x = 0
if x < 2 :
    print('малий')
elif x < 10 :
    print('Середній')
else :
    print('ВЕЛИКИЙ')
print('Все зроблено')
```



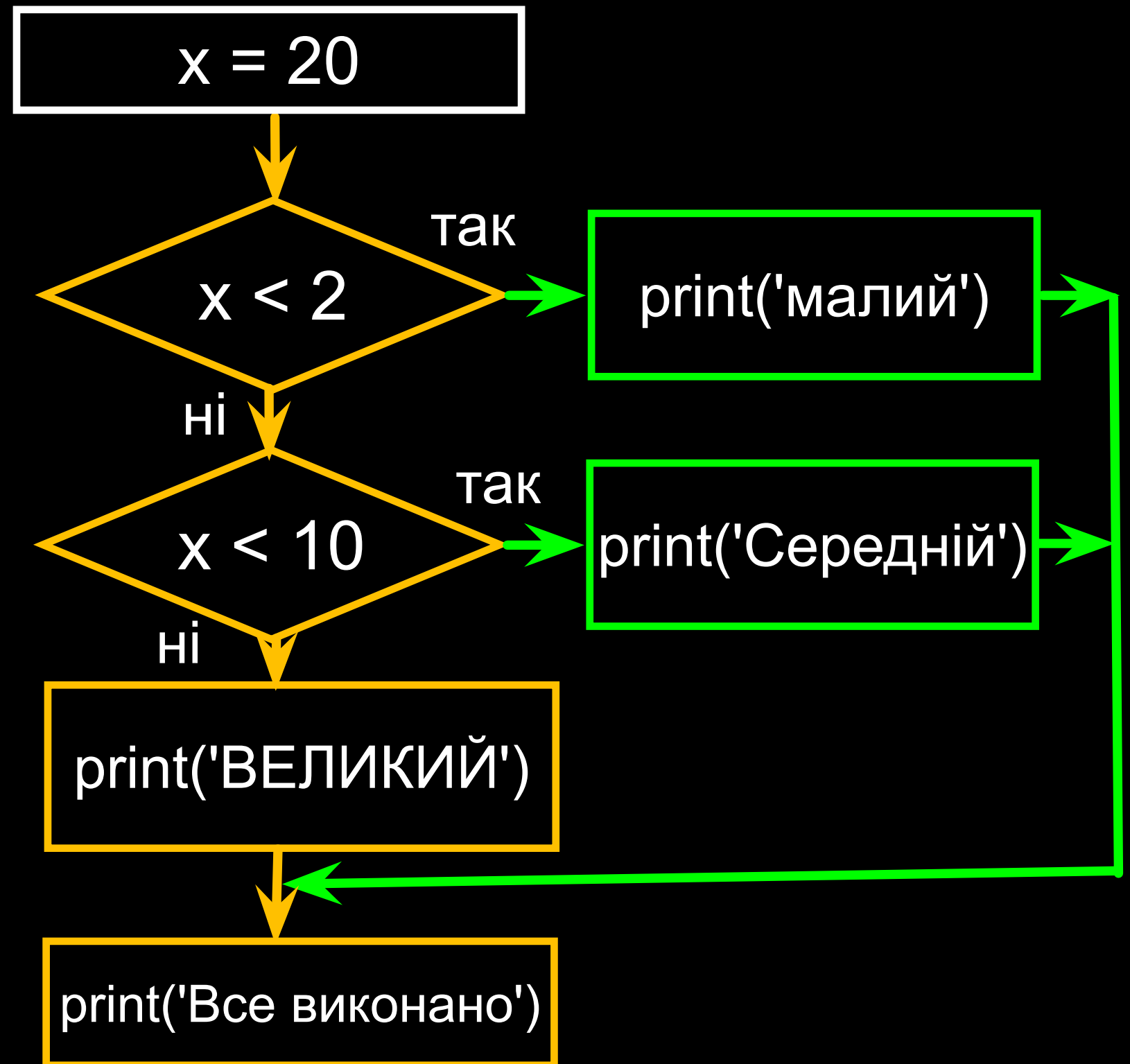
Множинне розгалуження

```
x = 5
if x < 2 :
    print('малий')
elif x < 10 :
    print('Середній')
else :
    print('ВЕЛИКИЙ')
print('Все зроблено')
```



Множинне розгалуження

```
x = 20
if x < 2 :
    print('малий')
elif x < 10 :
    print('Середній')
else :
    print('ВЕЛИКИЙ')
print('Все виконано')
```



Множинне розгалуження

```
# Без Else
x = 5
if x < 2 :
    print('Маленьке')
elif x < 10 :
    print('Середнє')

print('Все зроблено')
```

```
if x < 2 :
    print('Маленьке')
elif x < 10 :
    print('Середнє')
elif x < 20 :
    print('Велике')
elif x < 40 :
    print('Дуже велике')
elif x < 100:
    print('Величезне')
else :
    print('Гігантське')
```

Множинні логічні задачі

Що ніколи не буде виводитись
незалежно від значення x?

```
if x < 2 :  
    print('Менше 2')  
elif x >= 2 :  
    print('Два або більше')  
else :  
    print('Щось інше')
```

```
if x < 2 :  
    print('Менше 2')  
elif x < 20 :  
    print('Менше 20')  
elif x < 10 :  
    print('Менше 10')  
else :  
    print('Щось інше')
```

Концепція try / except

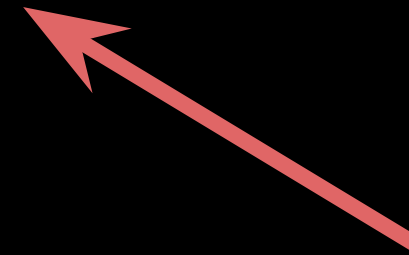
- «Небезпечний» код ви поміщаєте у вирази `try` та `except`
- Якщо код в `try` працює — код в `except` пропускається
- Якщо код в блоці `try` видає помилку, програма переходить до коду у блоці `except`

```
$ cat notry.py
astr = 'Hello Bob'
istr = int(astr)
print('First', istr)
astr = '123'
istr = int(astr)
print('Second', istr)
```

```
$ python3 notry.py
```

```
Traceback (most recent call last):
File "notry.py", line 2, in <module>
istr = int(astr)ValueError: invalid literal
for int() with base 10: 'Hello Bob'
```

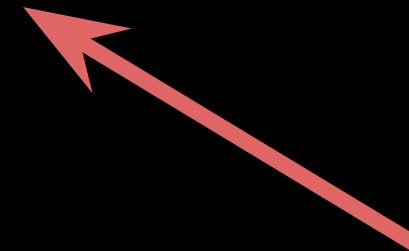
Все
ГОТОВО



```
$ python3 notry.py
```

```
Traceback (most recent call last):  
File "notry.py", line 2, in <module>  
istr = int(astr)ValueError: invalid literal  
for int() with base 10: 'Hello Bob'
```

Все
ГОТОВО

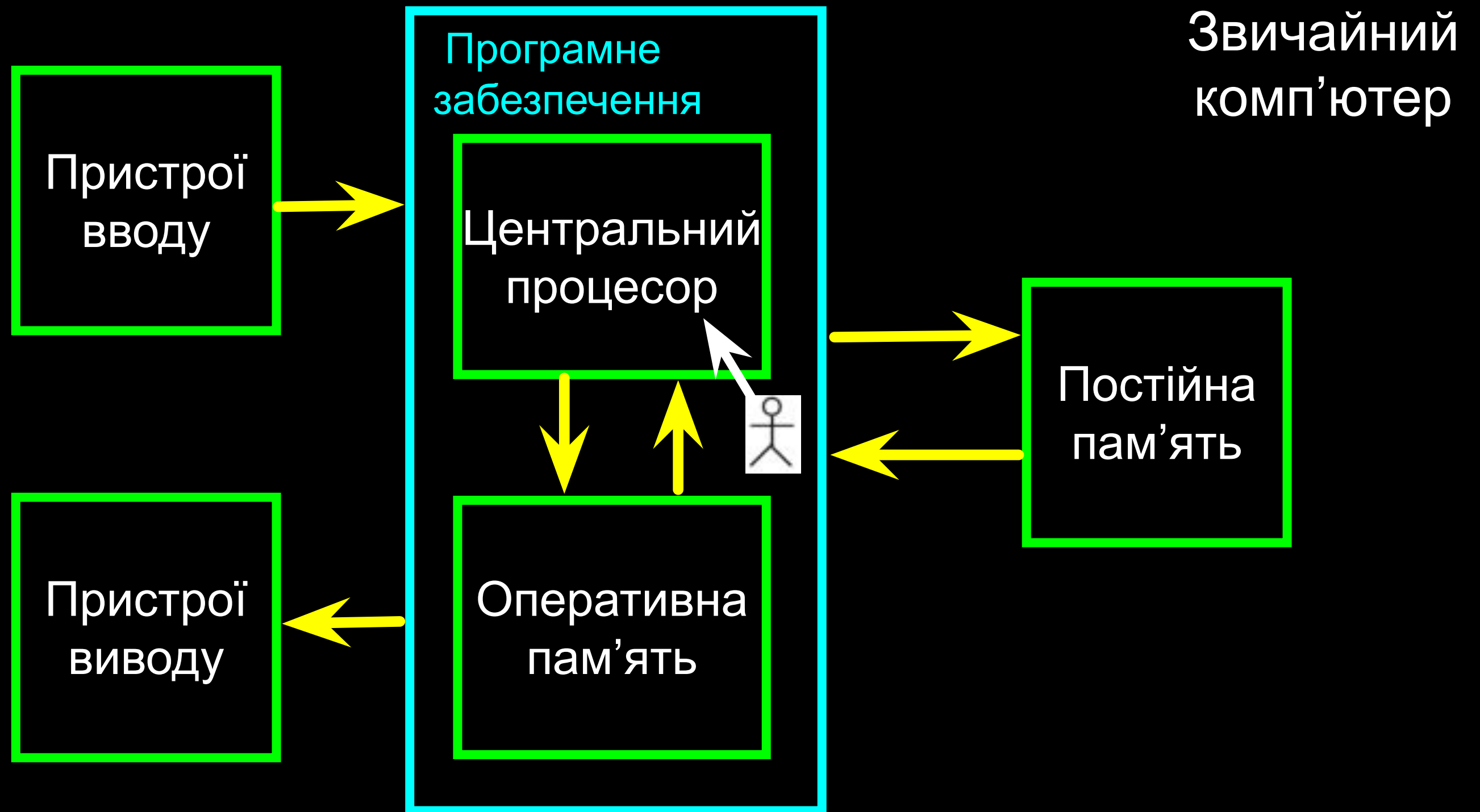


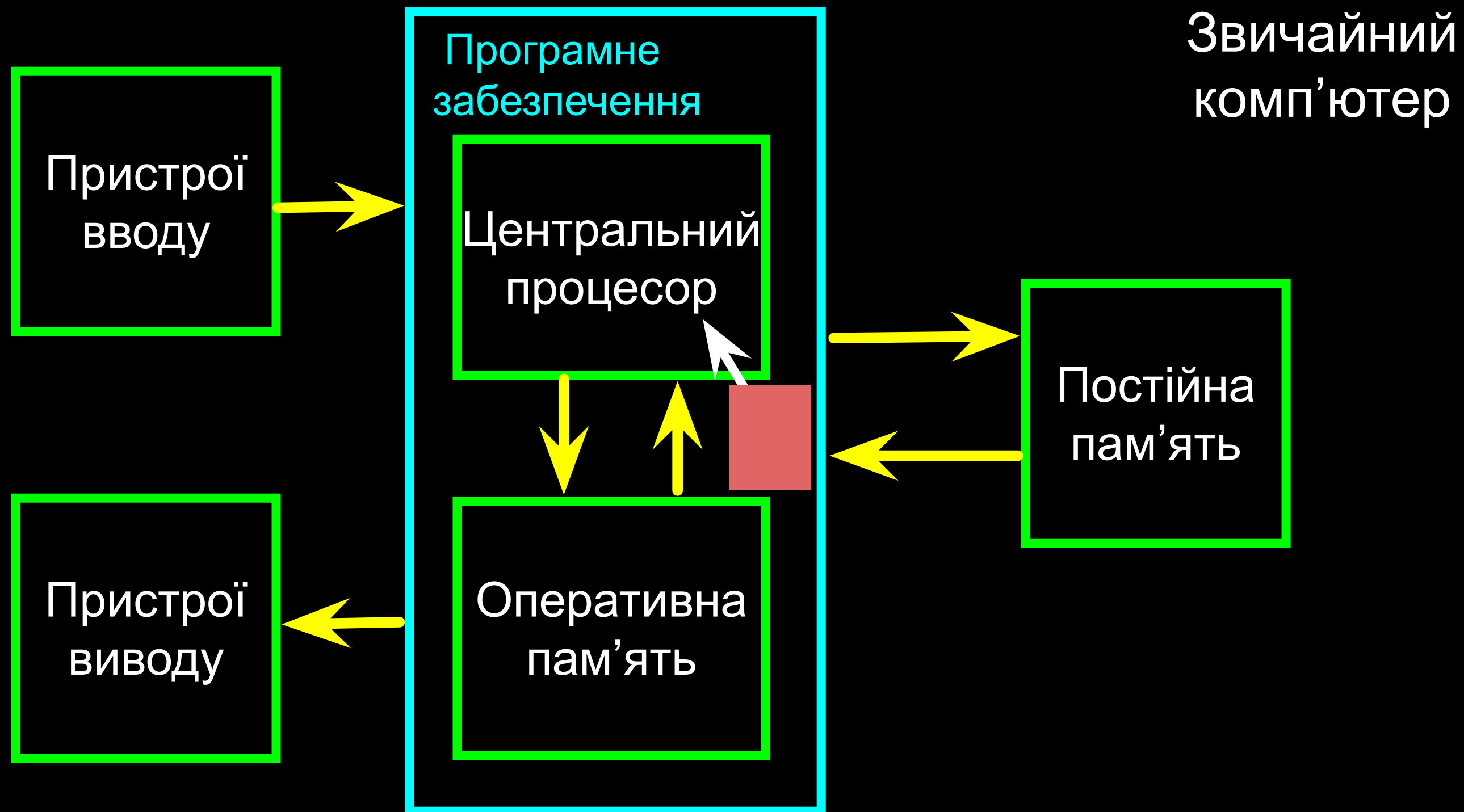
На цьому
програма
зупиняється



```
$ cat notry.py  
astr = 'Hello Bob'  
istr = int(astr)
```








```
astr = 'Hello Bob'
try:
    istr = int(astr)
except:
    istr = -1
print('First', istr)
```

```
astr = '123'
try:
    istr = int(astr)
except:
    istr = -1
print('Second', istr)
```

Якщо при першій конвертації щось
йде не так, буде запущено лише
інструкції в блоці `except`, і програма
йде далі

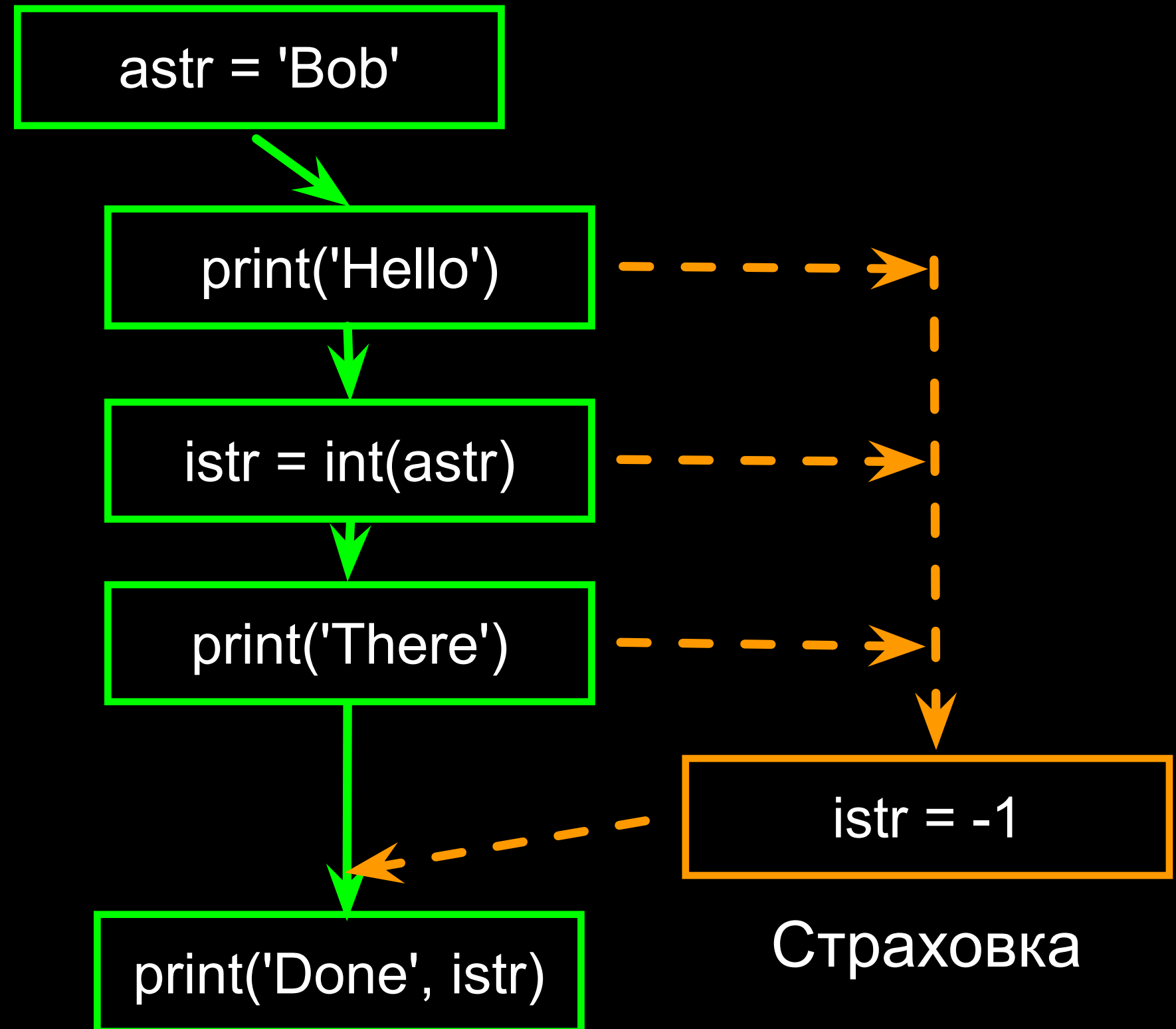
```
$ python tryexcept.py
First -1
Second 123
```

Якщо друга конвертація успішна
– `except` пропускається, і
програма продовжує роботу

try / except

```
astr = 'Bob'
try:
    print('Hello')
    istr = int(astr)
    print('There')
except:
    istr = -1

print('Done', istr)
```



Приклади try / except

```
rawstr = input('Enter a number:')
try:
    ival = int(rawstr)
except:
    ival = -1

if ival > 0 :
    print('Nice work')
else:
    print('Not a number')
```

```
$ python3 trynum.py
Enter a number:42
Nice work
$ python3 trynum.py
Enter a number:forty-two
Not a number
$
```

Підсумки

- Оператори порівняння
`==` `<=` `>=` `>` `<` `!=`
- Відступи
- Умови з одним шляхом
- Умови з двома шляхами:
`if:` та `else:`
- Вкладені умови
- Множинне розгалуження з
`elif`
- `try` / `except` для виключення помилок

Вправа

Перепишіть вашу програму оплати праці із застосуванням try і exsert так, щоб за години, відпрацьовані понад 40 годин, працівник отримував 1,5-кратну погодинну ставку.

Введіть години: 45

Введіть ставку: 10

Зарплата: 475.0

$$475 = 40 * 10 + 5 * 15$$

Вправа

Перепишіть вашу програму оплати праці із застосуванням try і exserpt так, щоб ваша програма коректно обробляла нечислові вхідні дані.

Введіть години: 20

Введіть ставку: дев'ять

Помилка, будь ласка, введіть числове значення

Введіть години: сорок

Помилка, будь ласка, введіть числове значення

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