Class 1

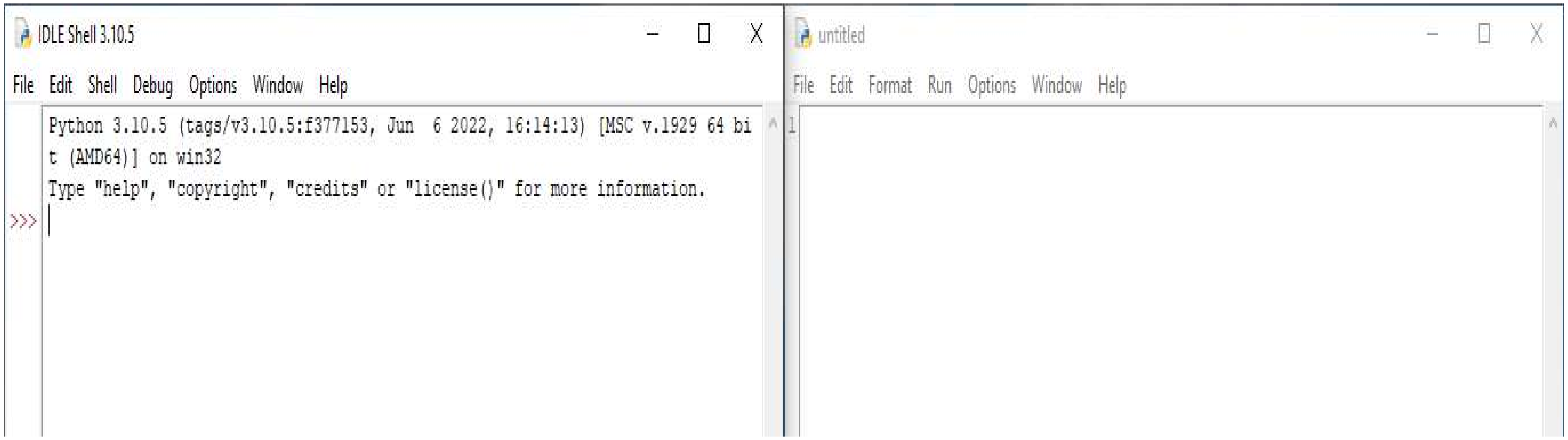
Introduction To Python

(Date: 7/28/2025)

Python Basic Concepts: (classified as High level language)

Interpreter Based Language	Case Sensitive Language
Indentation and whitespace	Supports Procedural and Object oriented Approaches

Interface Details


IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
untitled
File Edit Format Run Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

Variables:

A Variable is a container for storing and manipulating data

Naming rules:

- ❑ Must starts with a letter or an underscore *(_)*.
- ❑ Cannot start with a number.
- ❑ Can only contain alphanumeric characters *(A-Z, a-z), (0-9)* and underscores *(_)*.
- ❑ Are case-sensitive (e.g. *age, Age* and *AGE* are distinct variables).
- ❑ Cannot be a Python Keyword (e.g. *if, else, while, for* etc.)

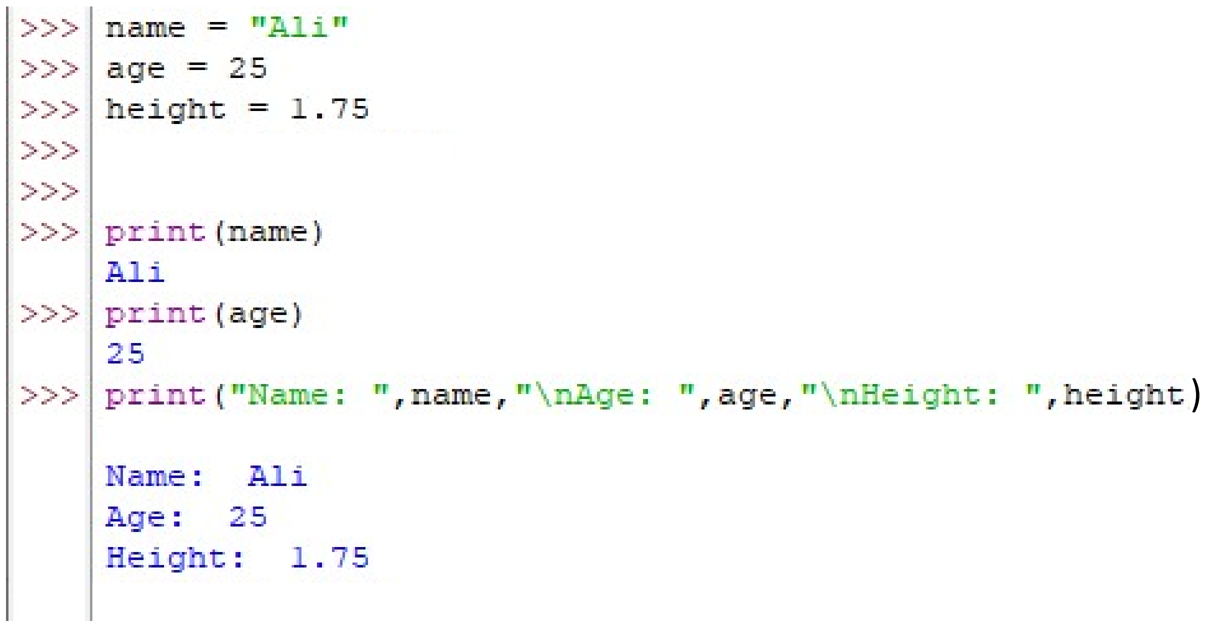
Variable:

Valid Python variables names

- string1
- _a1p4a
- list_of_names

Invalid Python variables names

- 9lives
- 99_balloons
- 2beOrNot2Be



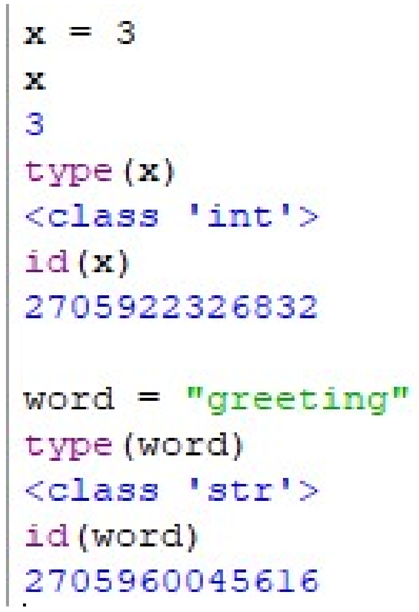
```
>>> name = "Ali"
>>> age = 25
>>> height = 1.75
>>>
>>>
>>> print(name)
    Ali
>>> print(age)
    25
>>> print("Name: ",name,"\nAge: ",age,"\nHeight: ",height)

    Name:  Ali
    Age:  25
    Height:  1.75
```

Python Memory Model:

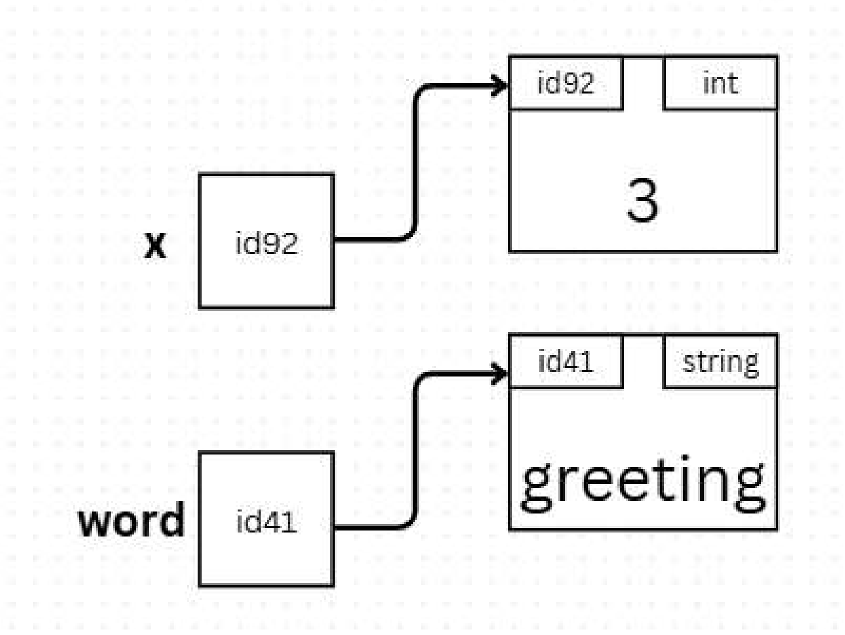
All data in Python program is stored in objects that have three components: *id, type, value*.

In Python, a variable is not an object, and so does not actually stores data; it stores an *id* that refers to an object that stores data



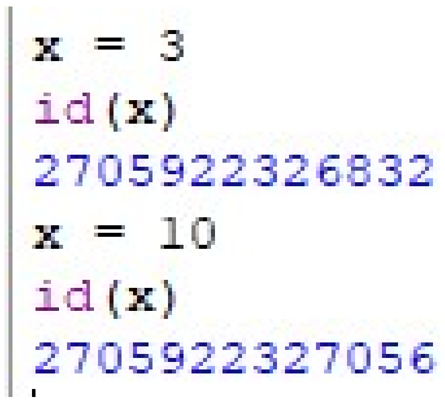
```
x = 3
x
3
type(x)
<class 'int'>
id(x)
2705922326832

word = "greeting"
type(word)
<class 'str'>
id(word)
2705960045616
```

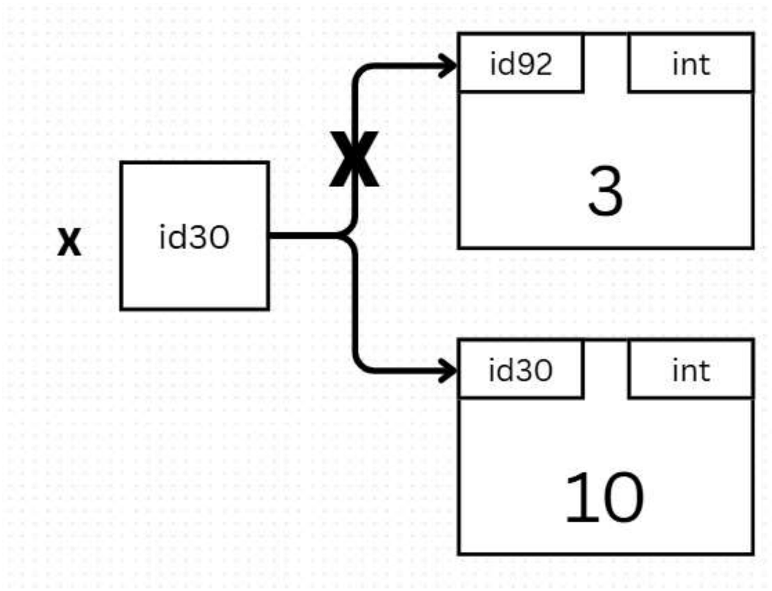


Python Memory Model:

Reassign a variable so it refers to the new object, which holds a new value



```
x = 3
id(x)
2705922326832
x = 10
id(x)
2705922327056
```



Python Memory model:

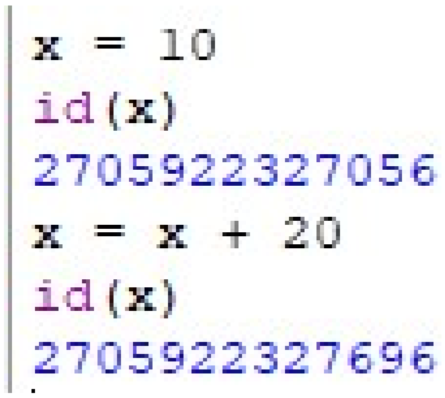
Executing an assignment statement

1. Evaluate the expression on R.H.S, yielding an *id* of the object.
2. If the variable on the L.H.S doesn’t exists, create it.
3. Store the *id* from the expression on the R.H.S in the variable on the L.H.S.

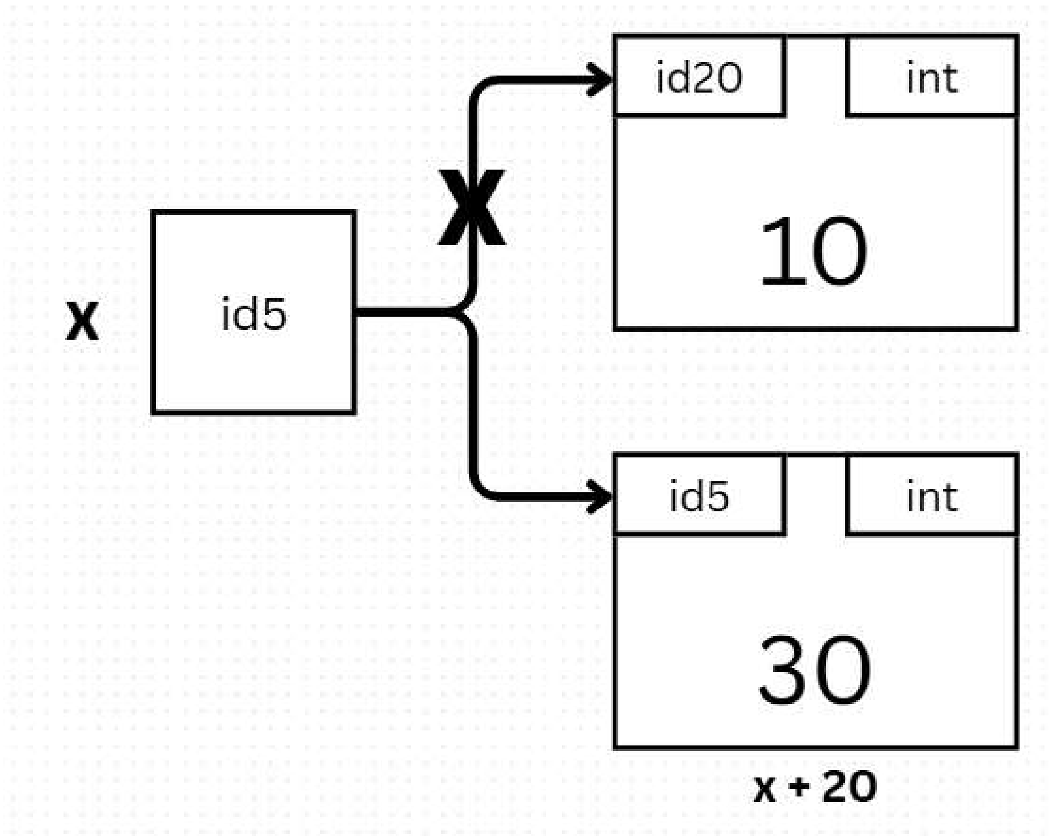
Evaluating an expression

1. If an expression is a variable, finds the variable. If it doesn’t exists, this is an error. If it does exists, the value of the expression is the *id* stored in that variable.
2. If the expression is a “literal value” such as *2, 175.6 or “hello”*, create an object for appropriate type to hold it. The value of the expression is the *id* of that object.
3. If an expression is an operator, such as ‘+’, evaluates its two operands, apply the operator to them, and creates a new object of the appropriate type to hold the result. The value of the expression is the *id* of that object.

Python Memory Model:



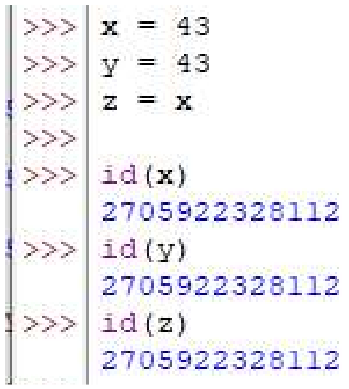
```
x = 10
id(x)
2705922327056
x = x + 20
id(x)
2705922327696
```



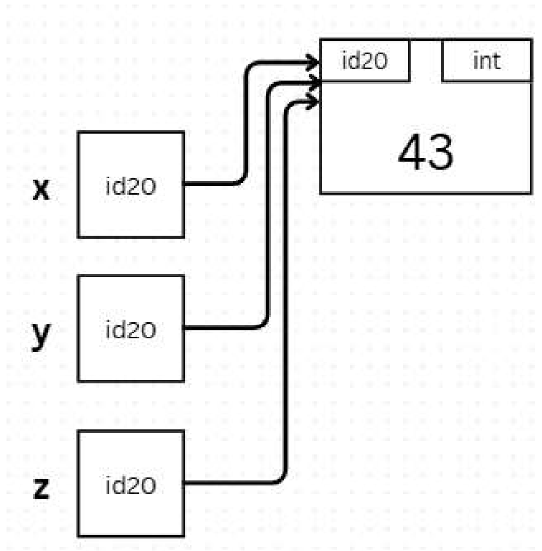
Python Memory Model:

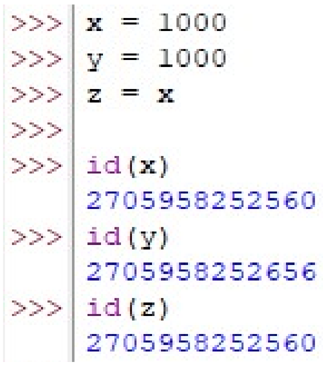
Python interning is due to some optimization and decision of Python core team.

The behavior in the below code happens to all small integers from *-5 to 128* and *to short strings*.

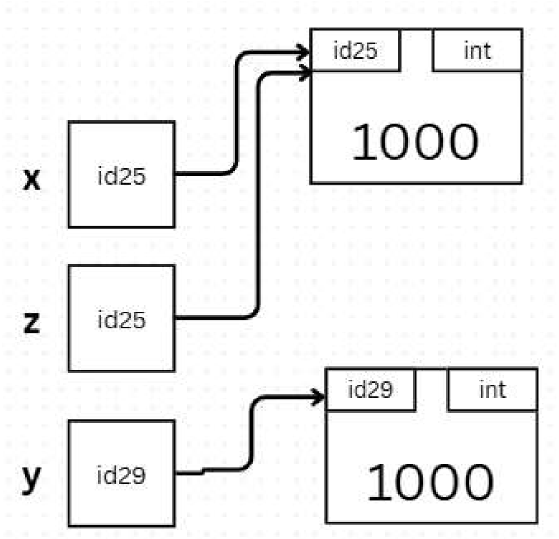


```
>>> x = 43
>>> y = 43
>>> z = x
>>>
>>> id(x)
2705922328112
>>> id(y)
2705922328112
>>> id(z)
2705922328112
```



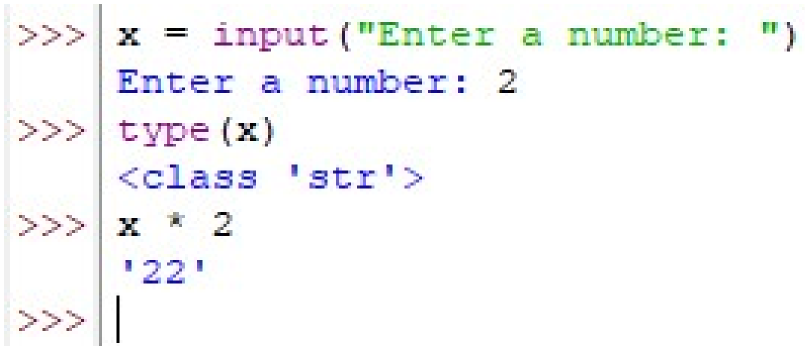


```
>>> x = 1000
>>> y = 1000
>>> z = x
>>>
>>> id(x)
2705958252560
>>> id(y)
2705958252656
>>> id(z)
2705958252560
```

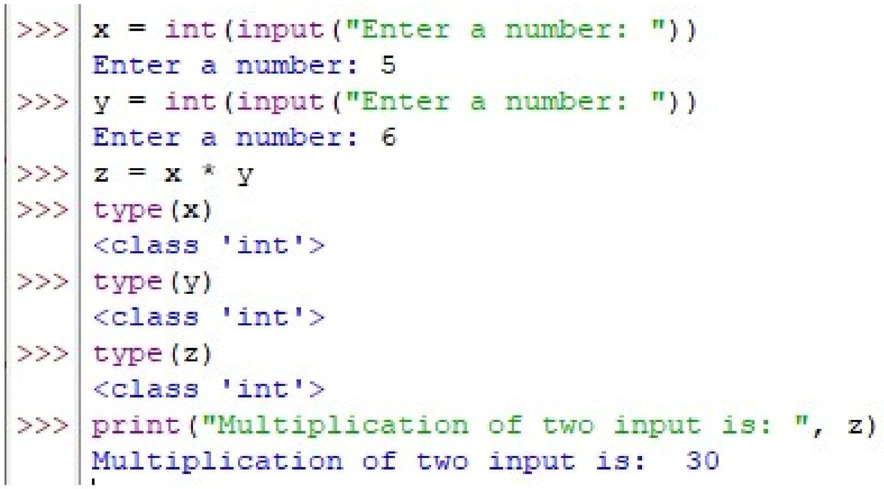


Data Types:

- Integers and float:



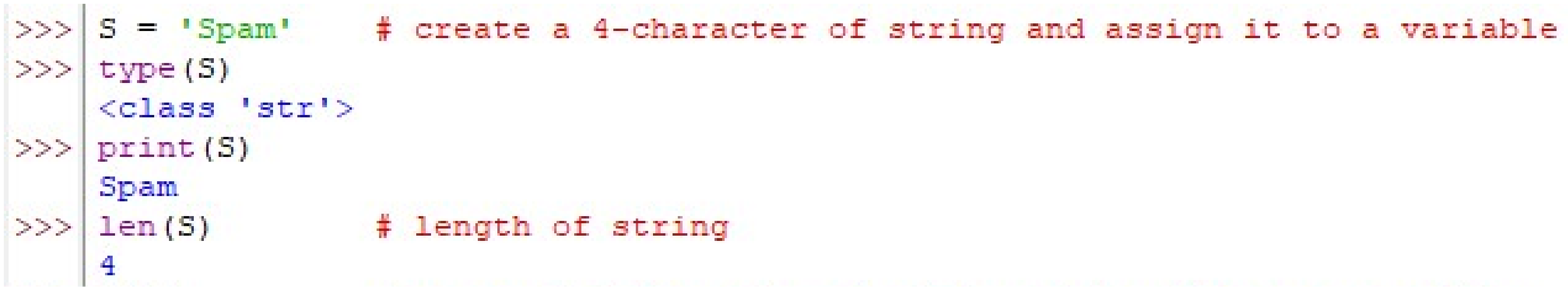
```
>>> x = input("Enter a number: ")
Enter a number: 2
>>> type(x)
<class 'str'>
>>> x * 2
'22'
>>>
```



```
>>> x = int(input("Enter a number: "))
Enter a number: 5
>>> y = int(input("Enter a number: "))
Enter a number: 6
>>> z = x * y
>>> type(x)
<class 'int'>
>>> type(y)
<class 'int'>
>>> type(z)
<class 'int'>
>>> print("Multiplication of two input is: ", z)
Multiplication of two input is:  30
```

Try this code with float function…
Syntax → float()

- **Strings:**
  - Any data types surrounded by quotation marks (either single or double) is termed as strings. It can be indexed positively and negatively.
  - Strings are immutable in Python i.e. they cannot be changed in place after they are created.



```
>>> S = 'Spam'     # create a 4-character of string and assign it to a variable
>>> type(S)
    <class 'str'>
>>> print(S)
    Spam
>>> len(S)         # length of string
    4
```

Operators:

- Operators are special symbols or keywords used to perform operation on one or more values (operands).

o Arithmetic assignment operators:

Operators	Example	Same as
= (Assignment)	x = 5	x = 5
+= (Add and assign)	x += 3	x = x + 3
-= (Subtract and assign)	x -= 3	x = x – 3
*= (Multiply and assign)	x *= 3	x = x * 3
/= (Divide and assign)	x /= 3	x = x / 3
%= (Modulus and assign)	x %= 3	x = x % 3
**= (Exponentiation and assign)	x **= 3	x = x ** 3
//= (Floor divide and assign)	x //= 3	x = x // 3