

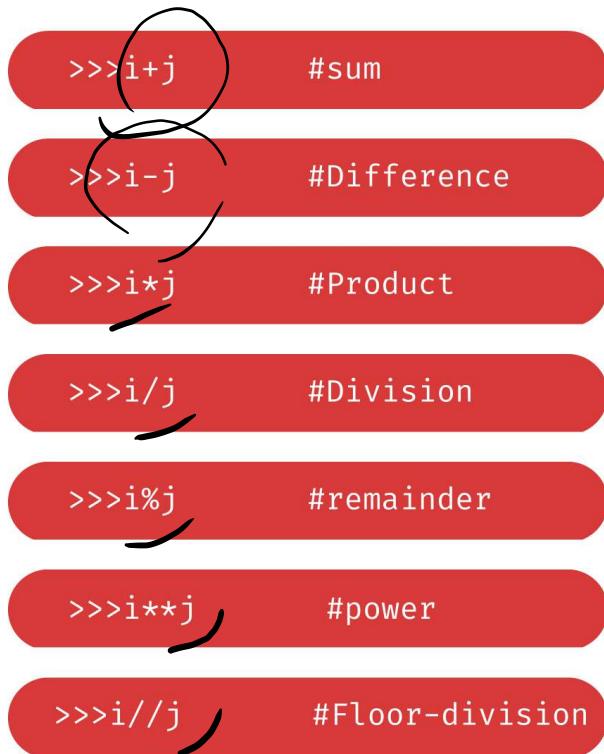


**Operators**

# Operators

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

# Arithmetic Ops.



are used with numeric values to  
perform common mathematical  
operations

# Assignment Ops.

x=5 # same as x=5

x+=3 # x= x+3

x-=3 # x= x-3

x\*=3 # x= x\*3

x/=3 # x= x/3

x%=3 # x= x%3

x//=3 # x= x//3

x\*\*=3 # x= x\*\*3

x&=3 # x= x&3

x|=3 # x= x|3

x^=3 # x= x^3

are used to assign values to variables



# Comparison Ops.

True

False

```
>>> i < j
```

#i, j: int, float, string!

```
>>> i >= j
```

```
>>> i < j
```

```
>>> i <= j
```

```
>>> i == j
```

#equality test, True if i is the same as j

```
>>> i != j
```

#inequality test, True if i not the same as j



are used to compare two values

# Logical Ops.

a, b → boolean.  
 $\neg$  not a → a: True →  $\neg a$ : False.

False ←  
`>>> not a` #True: if a is False  
#False: if a is True

`>>> a and b` #True: if both are True

`>>> a or b` #True: if either or both are True

True. →

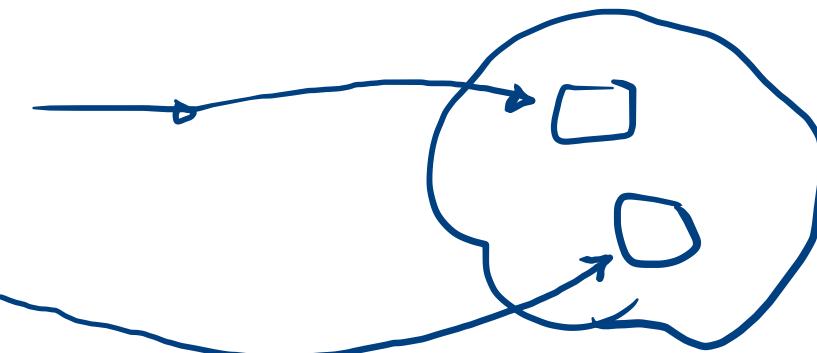
are used to combine conditional statements

a	b	a AND b	a OR b
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

$=$  vs  $\equiv$

$A = \underline{12}$

$B = 12$



$==$  vs  $=$  -->  
Comparing → assignment

$A \equiv B \rightarrow \underline{\text{False}}$

$A == B \rightarrow \text{True}$

Tree = 30

Tree  $==$  30  
check

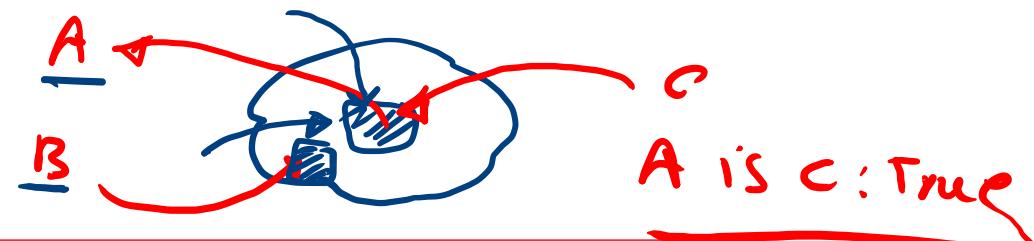
$A \text{ is } B \rightarrow \{$  False.

A is not B: True

A is not C: False

is #Returns True if both variables are the same object

is not #Returns True if both variables are not the same object



## Identity

### Ops.

A      B      ↓

are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location.

[list]

A: ['a', 'b', 'c']

## Membership Ops.

B = 'a'

B in A → True

nonscalar.

in #Returns True if a sequence with the specified value is  
#present in the object

not in #Returns True if a sequence with the specified value is  
#not present in the object

are used to test if a sequence is presented in an object

# Bitwise Ops.

- & #AND, Sets each bit to 1 if both bits are 1
- | #OR, Sets each bit to 1 if one of two bits is 1
- ^ #XOR, Sets each bit to 1 if only one of two bits is 1
- ~ #NOT, Inverts all the bits

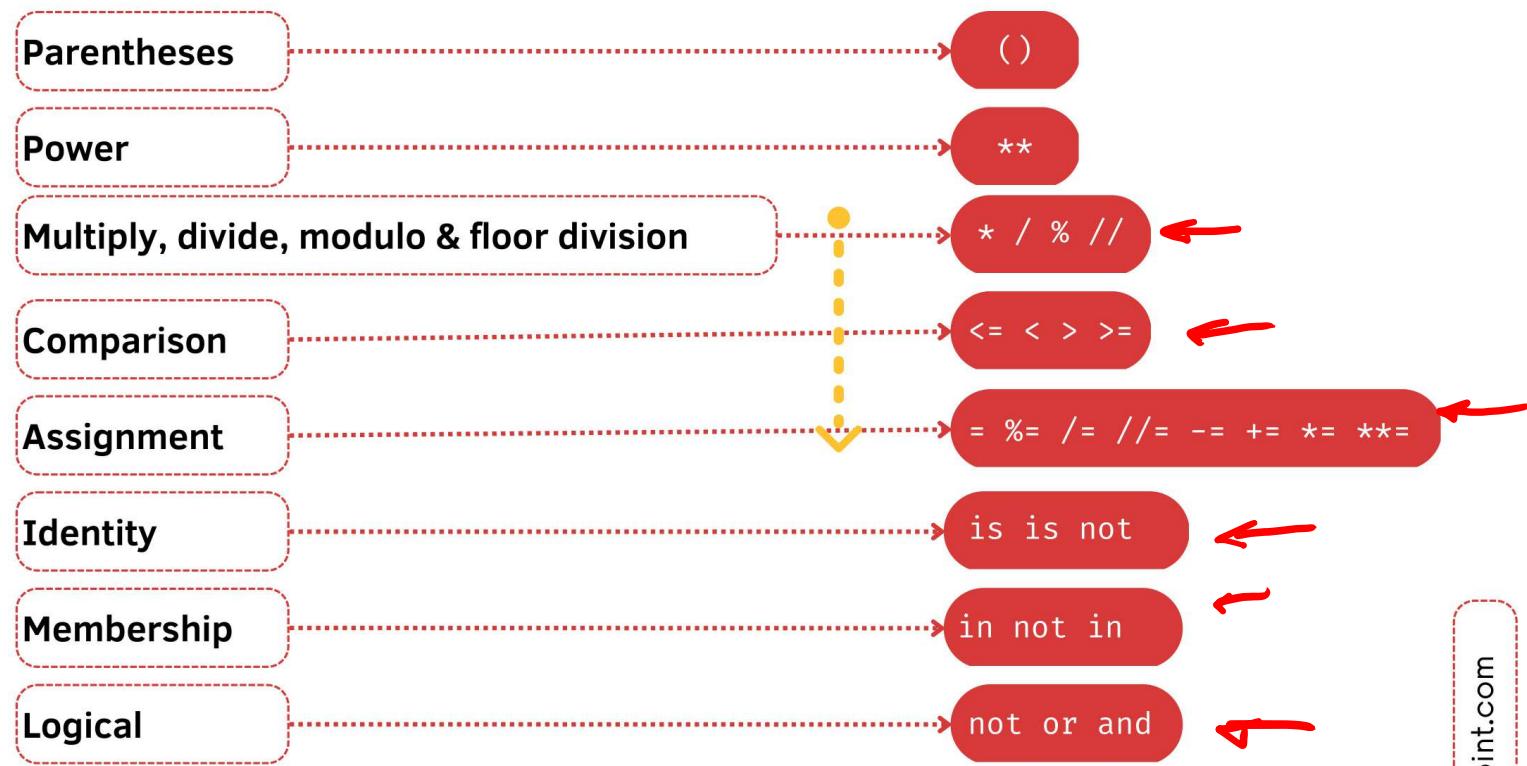


are used to compare (binary) numbers

$$3 * 2 - 5 / 18$$

( )

# Operator Precedence



# Operator Precedence

```
a = 20
b = 10
c = 15
d = 5
e = 0

e = (a + b) * c / d      #( 30 * 15 ) / 5
print ("Value of (a + b) * c / d is ", e)
```

## = VS ==

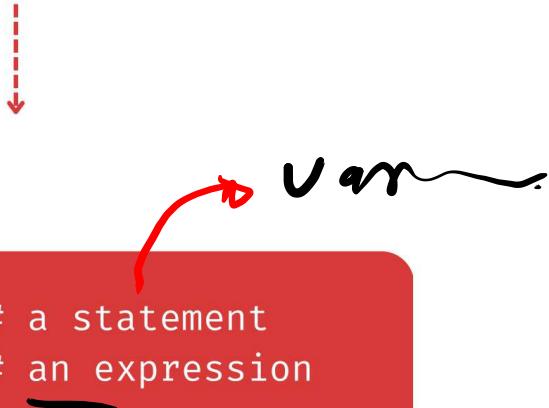
```
>>> x = float(input("Enter a number for x: "))
>>> y = float(input("Enter a number for y: "))
>>> if x == y:
    print("x and y are equal")
    if y != 0:
        print("therefore, x / y is", x/y)
    elif x < y:
        print("x is smaller")
    else:
        print("y is smaller")
>>> print("thanks!")
```

what if  $x = y$  ?

- combine objects and operators to form expressions
- an expression has a value, which has a type
- syntax for a simple expression: <object> <operator> <object>

# Expressions

```
x = 25      # a statement  
x = x + 10  # an expression
```



# Print

```
print(3+2)
```

#out:5

```
i=1
```

```
print(i)
```

#out:1

```
mark=18
```

```
std_name = 'Amin'
```

```
print('Mark of ' + std_name + ' ' + 'is: ' + mark)
```

#out: TypeError: can only concatenate str (not "int") to str

```
mark=18
```

```
std_name = 'Amin'
```

```
print('Mark of ' , std_name , ' ' , 'is: ' , mark)
```

#out: Mark of Amin is: 18

```
mark=18
```

```
mark_str = str(mark)
```

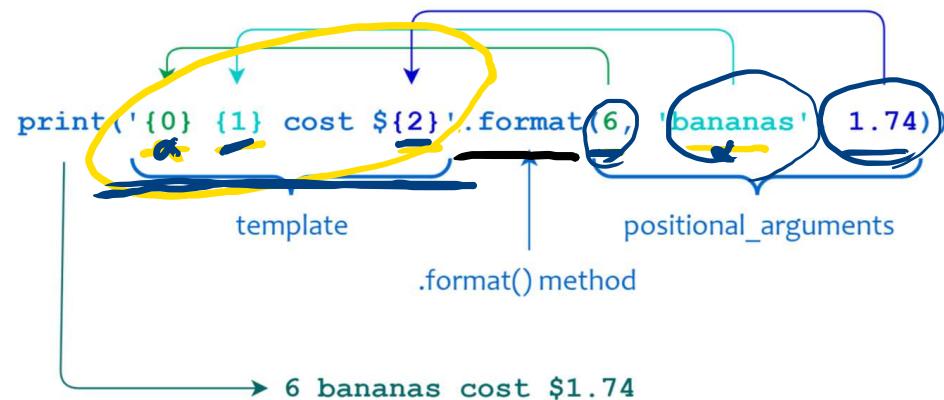
```
std_name = 'Amin'
```

```
print('Mark of ' + std_name + ' ' + 'is: ' + mark_str)
```

#out: Mark of Amin is: 18

to show output from code to a user, use print command

# Print (using format)



```
print('{1} {0}'.format('one', 'two'))  
#out: two one
```

# Binding Variables & Values

- equal sign (=) is an assignment of a value to a variable name
- value stored in computer memory
- an assignment binds name to value
- retrieve value associated with name or variable by invoking the name, by typing mark



```
x = 25          # a statement
x = x + 10      # an expression
```

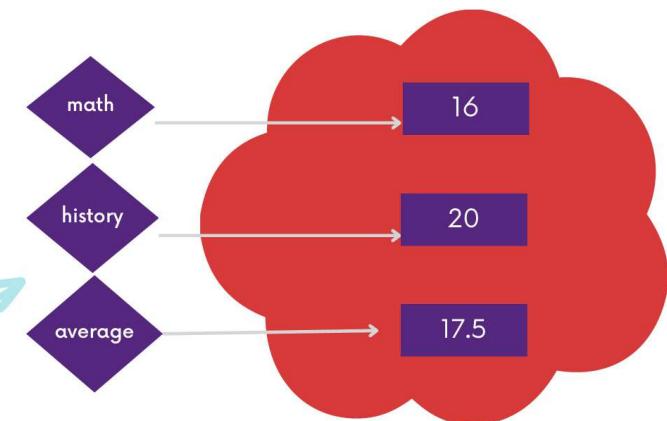
# Abstracting Expressions

## Abstracting Expressions

- to reuse names instead of values
- easier to change code later



```
math = 15  
history = 20  
average = (math + history)/2
```

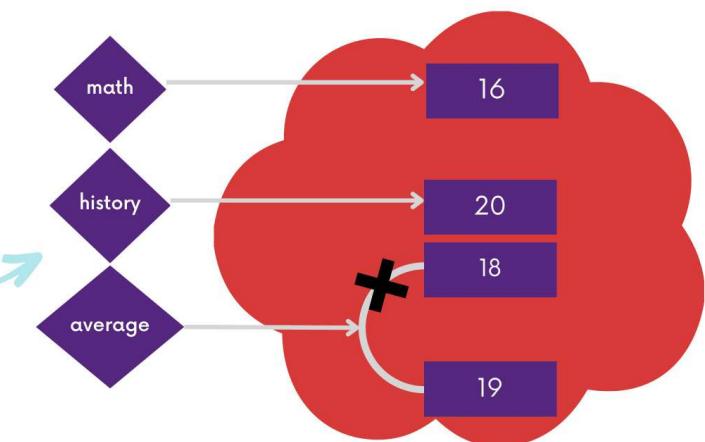


- Can **re-bind variable names** using new assignment statements.
- Previous value may still stored in memory but lost the handle for it
- Value for area does not change until you tell the computer to do the calculation again.

# Changing Binding



```
math = 16
history = 20
average = (math + history)/2
average = average + 1
```



# Strings

- Sequence of characters, Letters, Special Characters, Spaces, Digits
- enclose in quotation marks or single quotes
- Concatenate Strings

```
hi = "hello there"
```

```
hi = "hello there"  
name = "ana"  
  
greet = hi + name  
#out: hello thereana  
  
greeting = hi + " " + name  
#out: hello there ana  
  
operat = hi + " " + name * 3  
#out: hello there anaanaana
```

# String Literals

**Single quotes ('')**

**Double quotes ("")**

**Triple single quotes ('''')**

**triple-double quotes ("""")**

```
s = 'This is a string'  
print(s)
```

```
s = "Another string using double quotes"  
print(s)
```

```
s = ''' string can span multiple line '''  
print(s)
```

```
s = """ string can span multiple line """  
print(s)
```

# Input

- Prints whatever is in the quotes
- User types in something and hits enter
- Binds that value to a variable



```
text = input("Type anything ...")
print(text)
#out: input text
```

- **input** gives you a **string** so must cast if working with numbers



```
num = int(input("Type a number ..."))
print(num)
#out: input num
```

# Exercises 02

Operators  
900

Operators

## Exercise 01

- Get a number as circle diameter, and calculate the circle area!

Points:

- Use **input method** to get circle diameter!
- Use **str.format()** method to print the output!
- Output must be same as following example, exactly!

Example: input: 18 , output (print): Circle area is 254.469

Operators

## Exercise 02

- Get width, height and length ,then calculate and print the surface area and volume of a cuboid!

Points:

- Use **input method** to get the inputs!
- Use **str.format()** method to print the output!
- Output must be same as following example, exactly!

Example: input1: 18, input2:4, input3:5 , output (print): Volume of cuboid is 360.00 and Surface area of cuboid is 364.000!

Operators

## Exercise 03

- input(2 number) and print sum, division, subtraction, multiplication

Points:

- Use **input method** to get circle diameter!
- Use **str.format()** method to print the output!
- Output must be same as following example, exactly!

Example: input\_1: 8, input\_2: 4 , output: sum = 12, division=2, subtraction=4, multiplication=32

Operators

## Exercise 04

- Get two string, concatenate them and print the result!

Points:

- insert a space between two strings.
- Use **input method** to get circle diameter!
- Use **str.format()** method to print the output!
- Output must be same as following example, exactly!

Example: input\_1: string\_1 , input\_2: string\_2 , output: result is "string\_1 string\_2"

Operators

## Exercise 05

- Write a Python program to convert Fahrenheit to Celcius.

Points:

- Use **input method** to get Fohrenheit!
- Use **str.format()** method to print the output!
- Output must be same as following example, exactly!

Example: input: 86 , output: 86 degree Fahrenheit is equal to 30.0 degree Celsius.

Operators

## Exercise 06 (Search!)

- Write a code to create following pattern!

```
#  
##  
###  
####  
#####  
######  
######
```