PYTHON

MahmoudAlipour.a@Gmail.Com
@AminAlipour_IT

filoger

Syllabus

S01
- Basics
- Operators

S02
- Control Flow & Iteration
- Function

S03
- List
- Tuples
- Set
- String

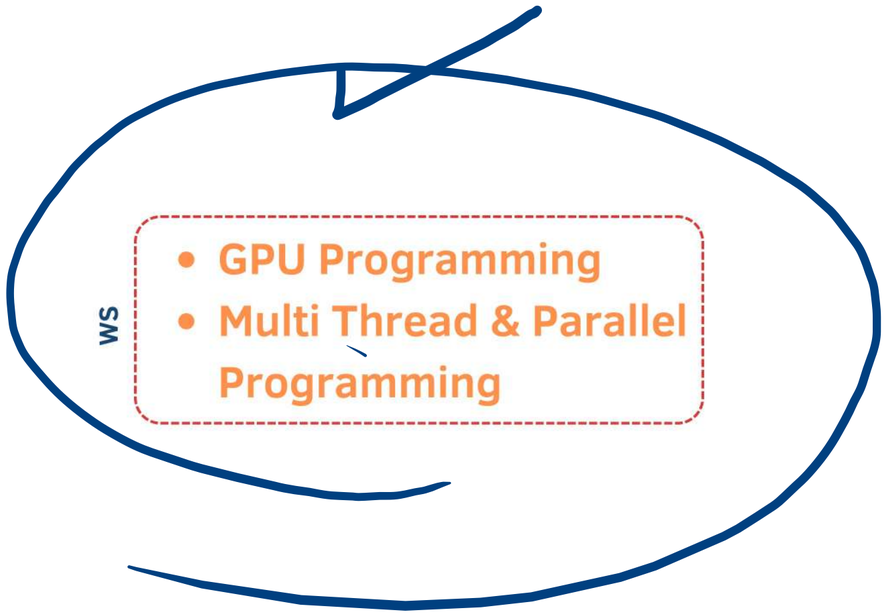
S04
- Exception Handling
- OOP (Object & Class)

S05
- Inheritance

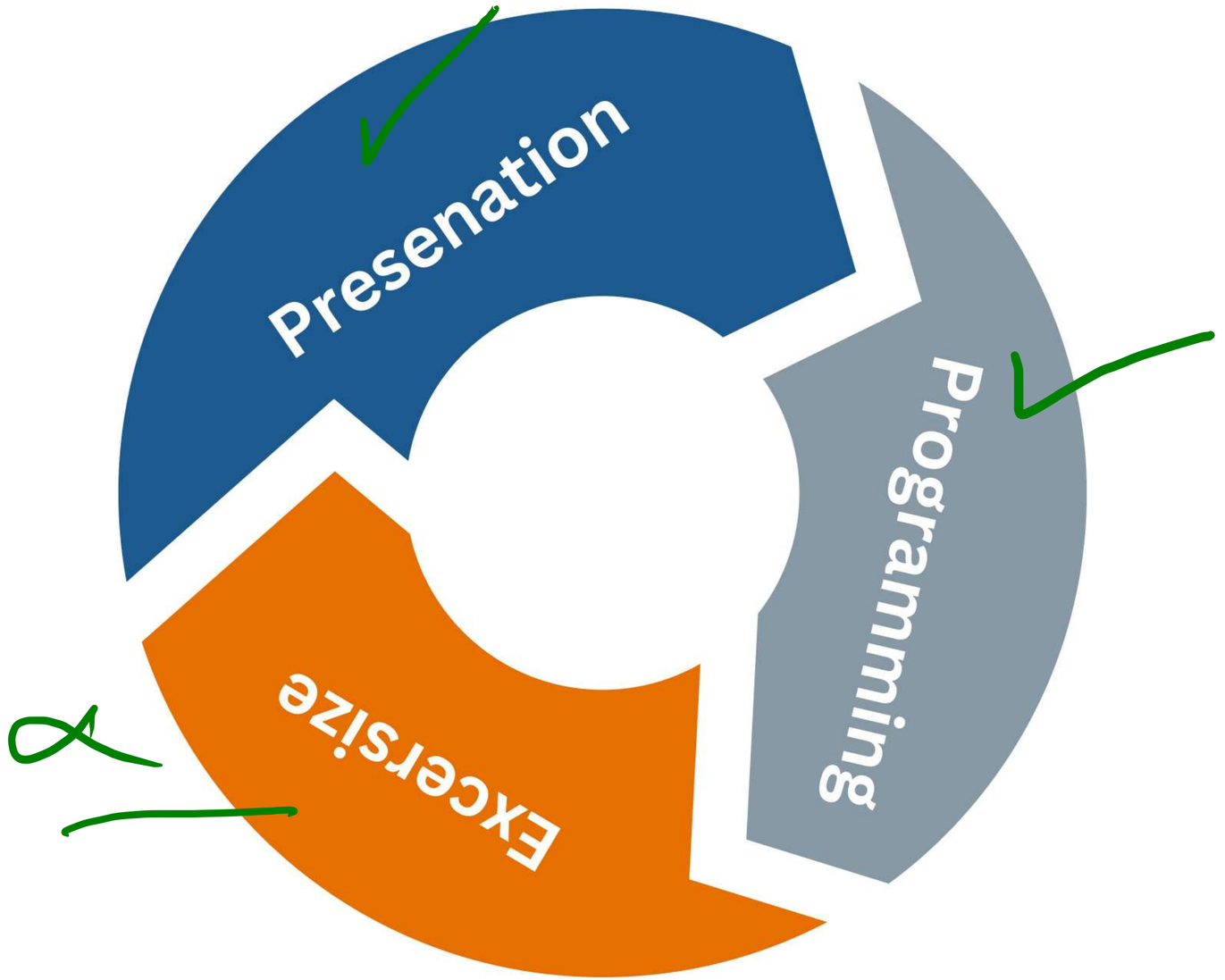
S06
- Abstraction
- Encapsulation

S07
- Numpy

S08
- Pandas

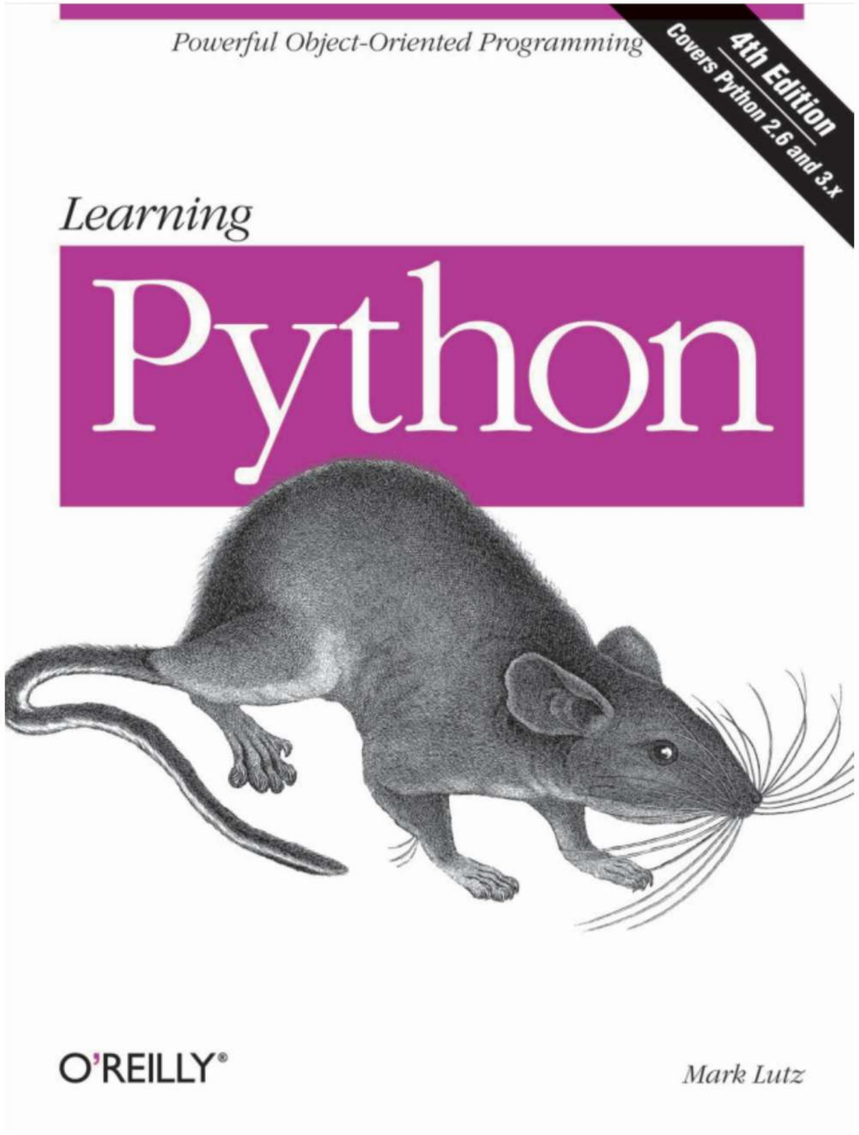


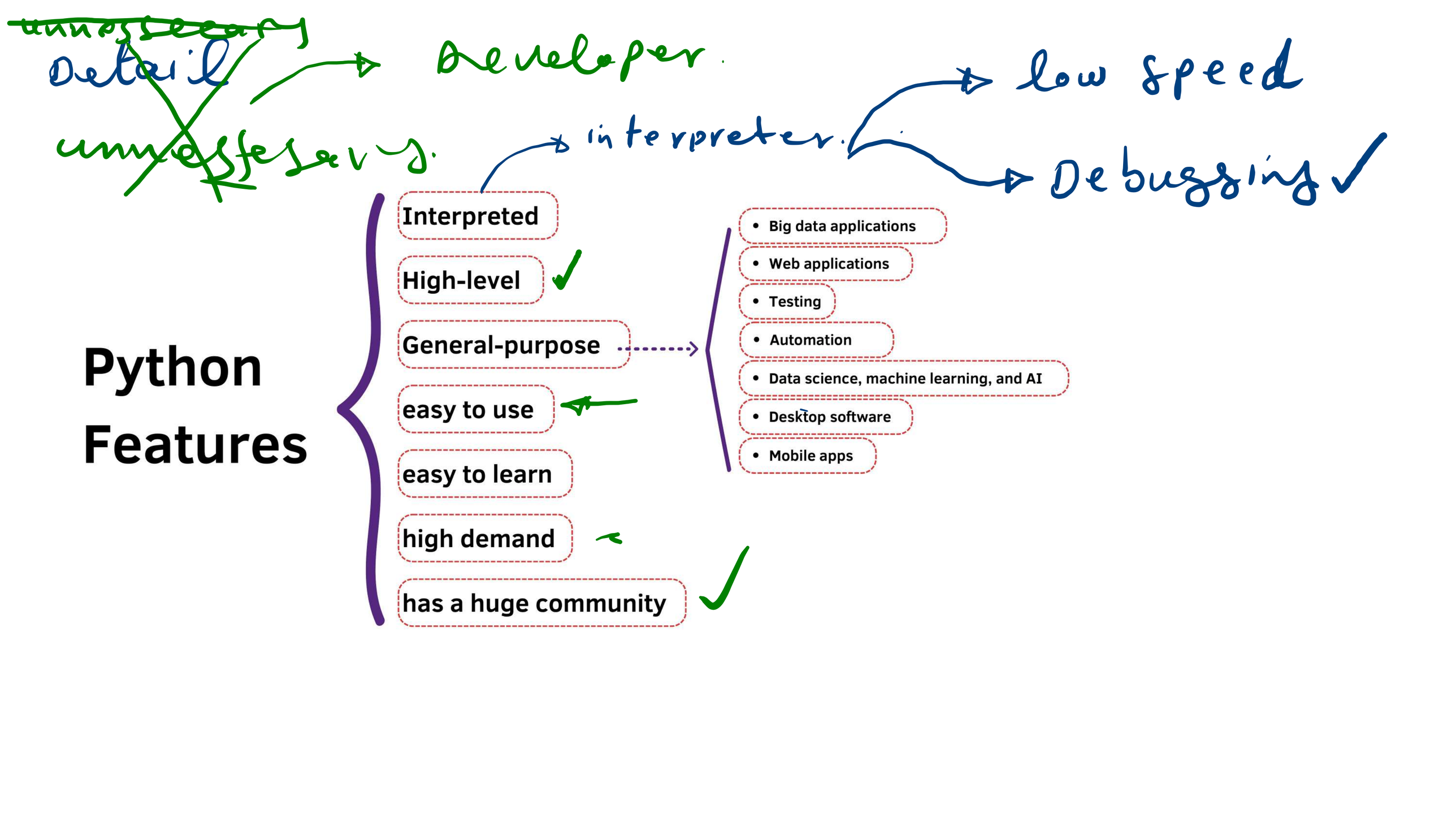
Teaching Methodology

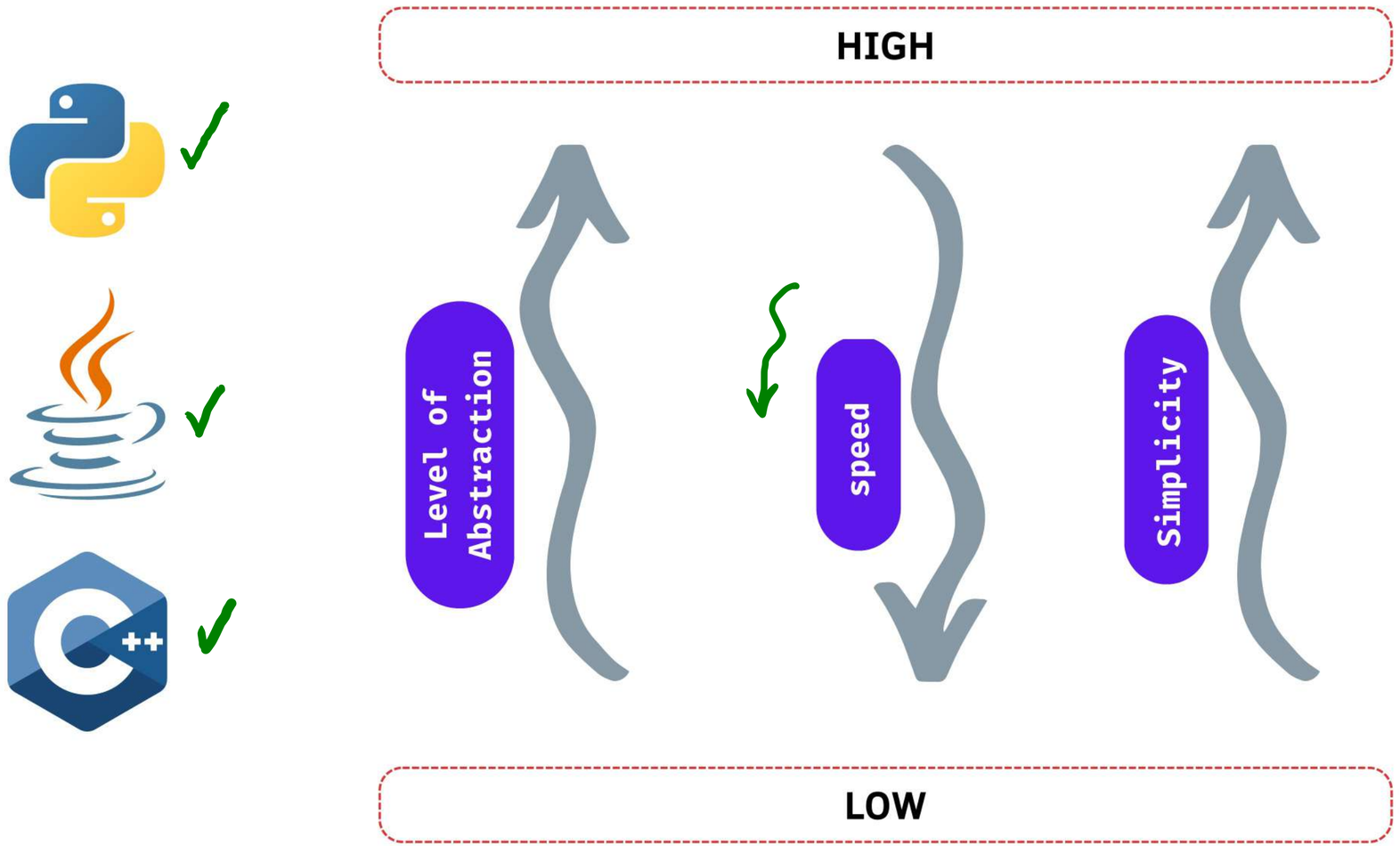


Ref.

- **Learning Python**
  Mark Lutz (4th Edition)
- **ocw.mit.edu**
- **pythontutorial.net**
- **github.com/Asabeneh**

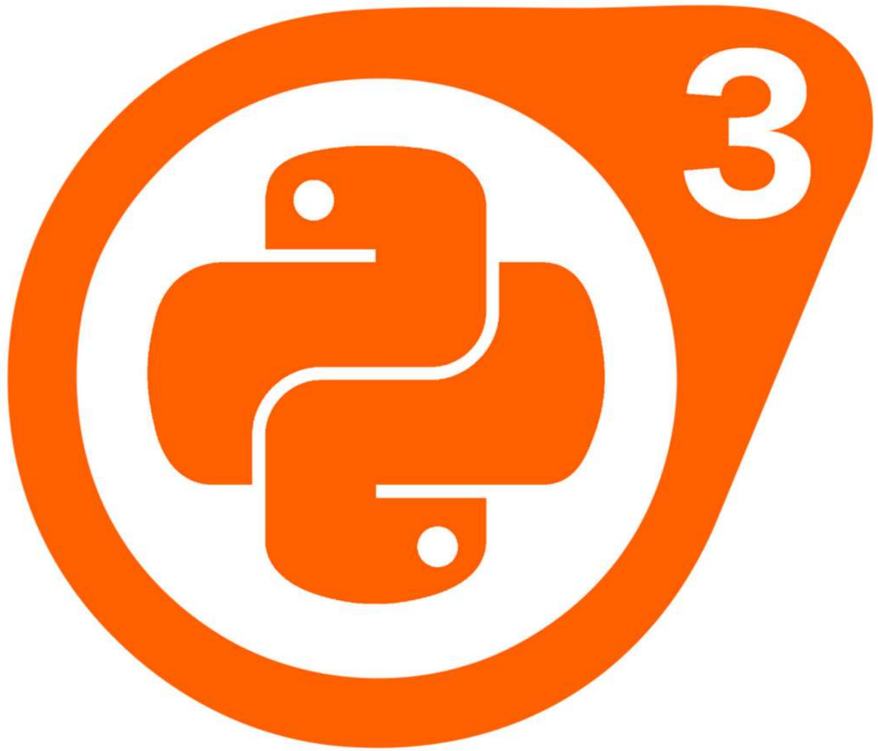



unnecessary
Detail
unnecessary.
Developer.
interpreter.
low speed
Debugging ✓
Python Features
Interpreted
High-level ✓
General-purpose
easy to use
easy to learn
high demand
has a huge community ✓
• Big data applications
• Web applications
• Testing
• Automation
• Data science, machine learning, and AI
• Desktop software
• Mobile apps



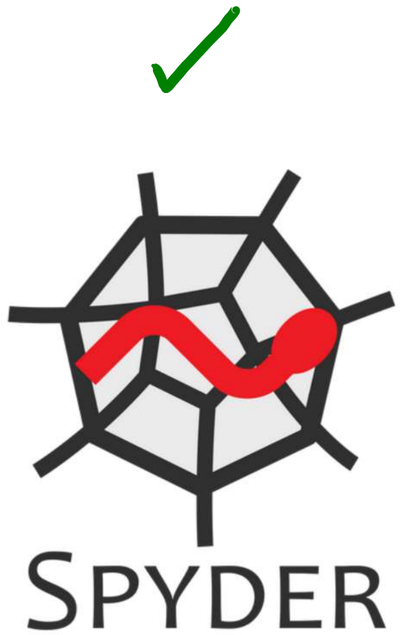
src: c-sharpcorner.com

Py2 | Py3 ?

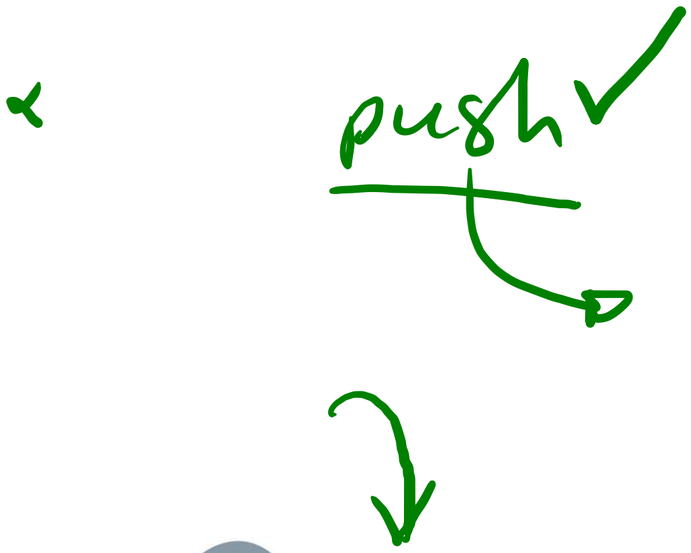


3.10

IDE



File Sharing



GitHub

https://github.com/MhmudAlpurd/Vision_Feb2023.git

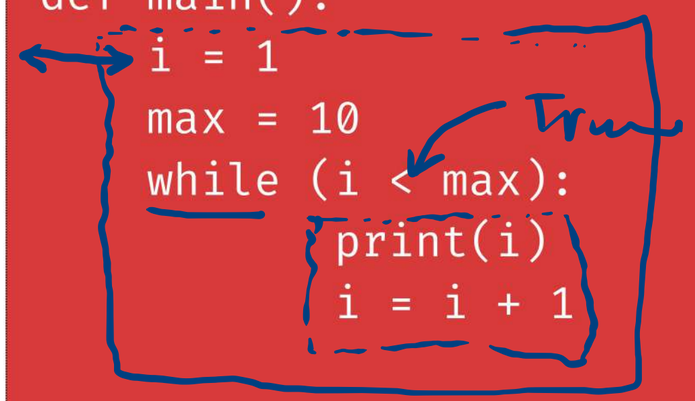
Basics

White space & Indentation

```
"""
define main function to
print out something
for test
"""

def main():
    i = 1
    max = 10
    while (i < max):
        print(i)
        i = i + 1

# call function main
main()
```



Comment

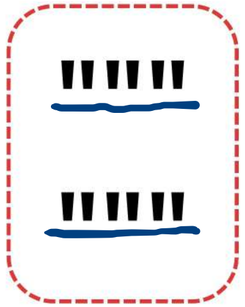
#

```
"""
define main function to
print out something
for test
"""

def main():
    i = 1
    max = 10
    while (i < max):
            print(i)
            i = i + 1

# call function main
main()
```

MultiLine Comment /DocString




```
"""
define main function to
print out something
for test
"""

def main():
    i = 1
    max = 10
    while (i < max):
            print(i)
            i = i + 1

# call function main
main()
```

Continuation of statements

\

```
if (a == True) and \
 (b == False) and (c == True):
     print("Continuation of
statements")
```

var-1 = 1.0

a-z
A-Z
/ -

1.
@ % ...

Identifiers

- Identifiers are names that identify variables, functions, modules, classes, and other objects in Python.
- The name of an identifier needs to begin with a letter or underscore (_). The following characters can be alphanumeric or underscore.
- Python identifiers are case-sensitive.

→ VAR-1 = 1.0

var-1 = 1.0

Keywords

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

Objects

almost

in Python everything is an Object, and Any Object has a Type.

Programs manipulate Data Objects.

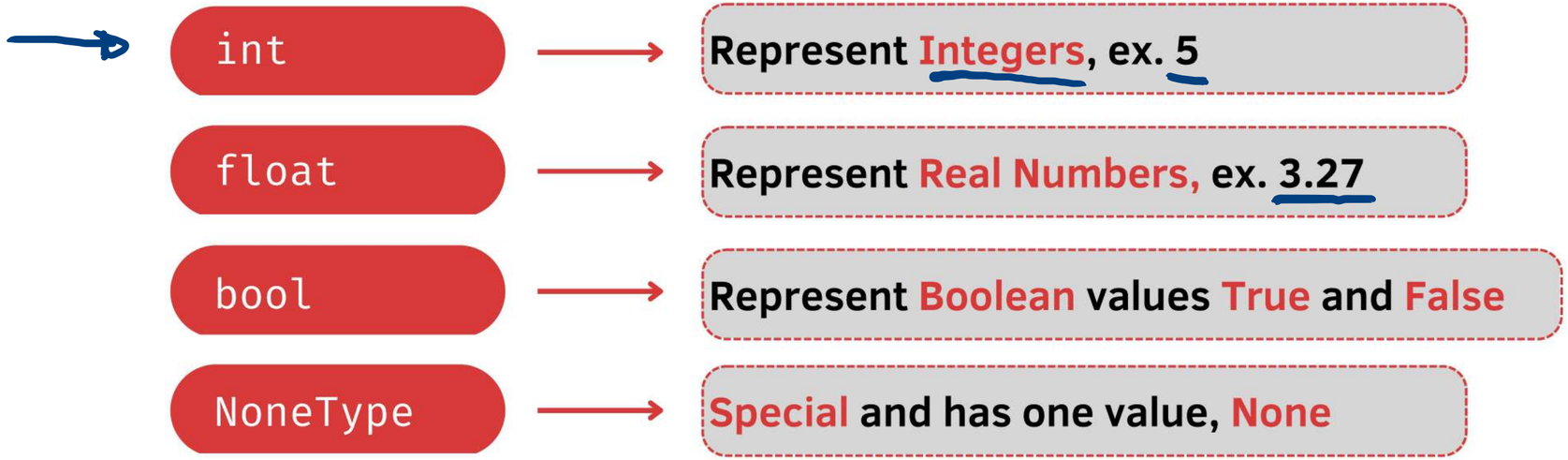
Objects have a Type that defines the kinds of things.

Objects are

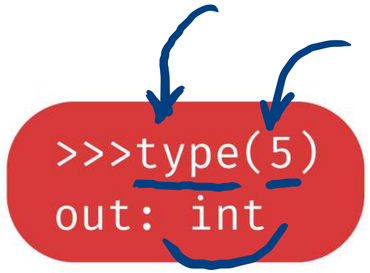
- Scalar (cannot be subdivided)
- Non-scalar (have internal structure that can be accessed)

Type → int
float
list
⋮

Scalar Objects



Type



Can use type() to see the type of an object

Double

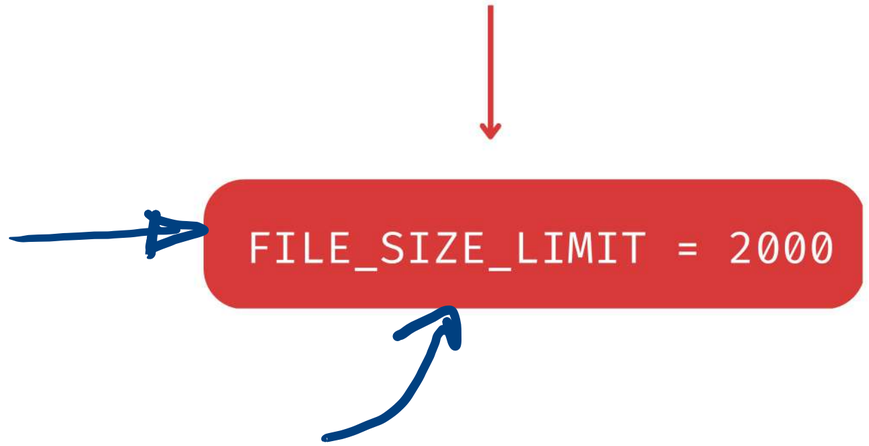
Python does not have an in-built double data type

-const

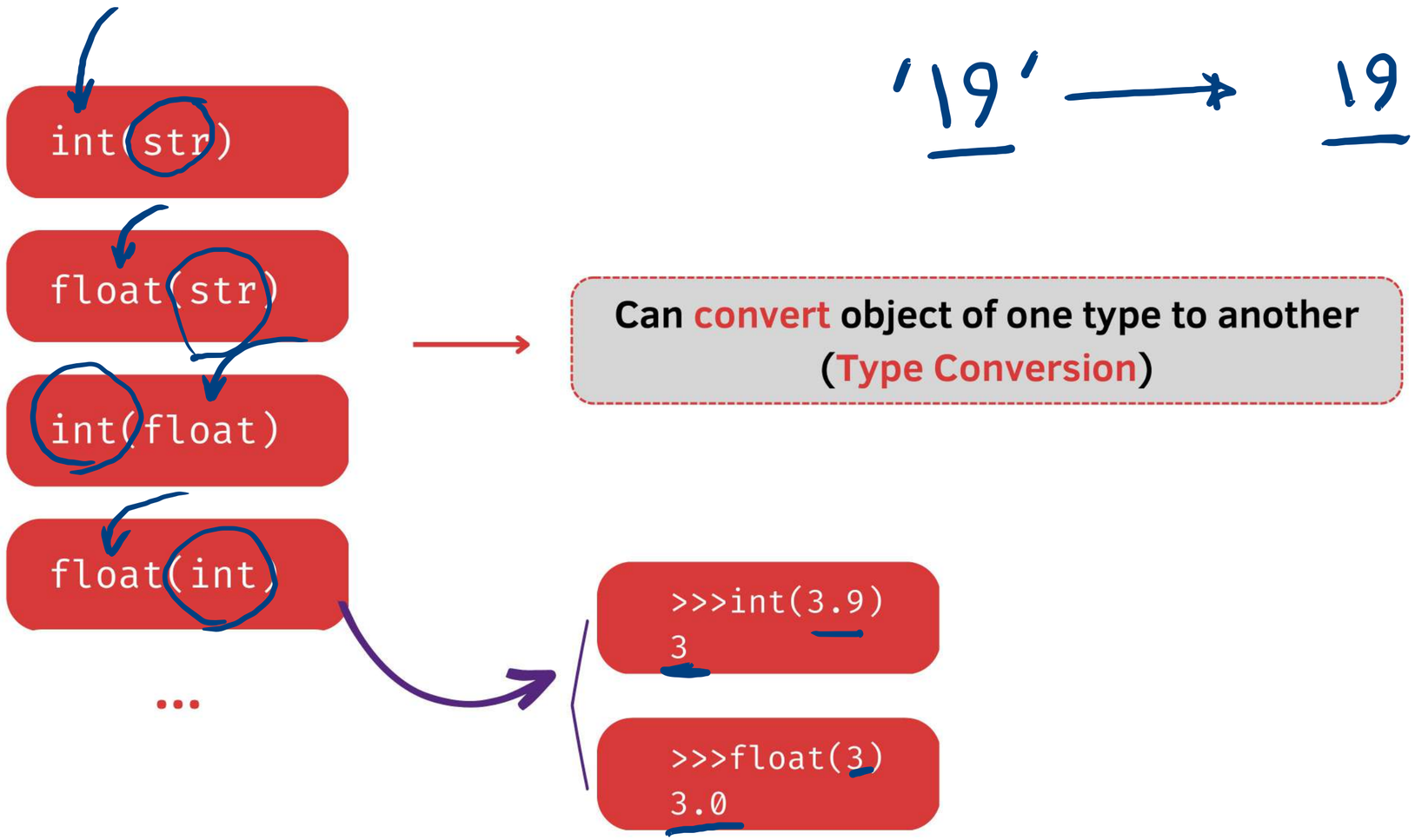
num_tree_const = 19

Constant

The bad news is that Python doesn't support constants.



Cast


int(str)
float(str)
int(float)
float(int)
...
Can convert object of one type to another
(Type Conversion)
>>>int(3.9)
3
>>>float(3)
3.0

Exercises 01

Basics

600

Basics

Exercise 01

- What is the type of the following data?

1. "Filoger"
2. "19.0"
3. 18
4. 13.0
5. True

ipynb

EXs 01

EX 01

EX .2

Basics

Exercise 02

int

- Convert the following data to integer?

1. "Filoger"
2. "19.0"
3. 18
4. 13.0
5. True

Basics

Exercise 03 (Search!)

- Print size of the following data in memory (bytes)?

1. "Filoger"
2. "19.0"
3. 18
4. 13.0
5. True

Basics

Exercise 04 (Search!)

- Print running time of program to find type of the following data ?

1. "Filoger"
2. "19.0"
3. 18
4. 13.0
5. True

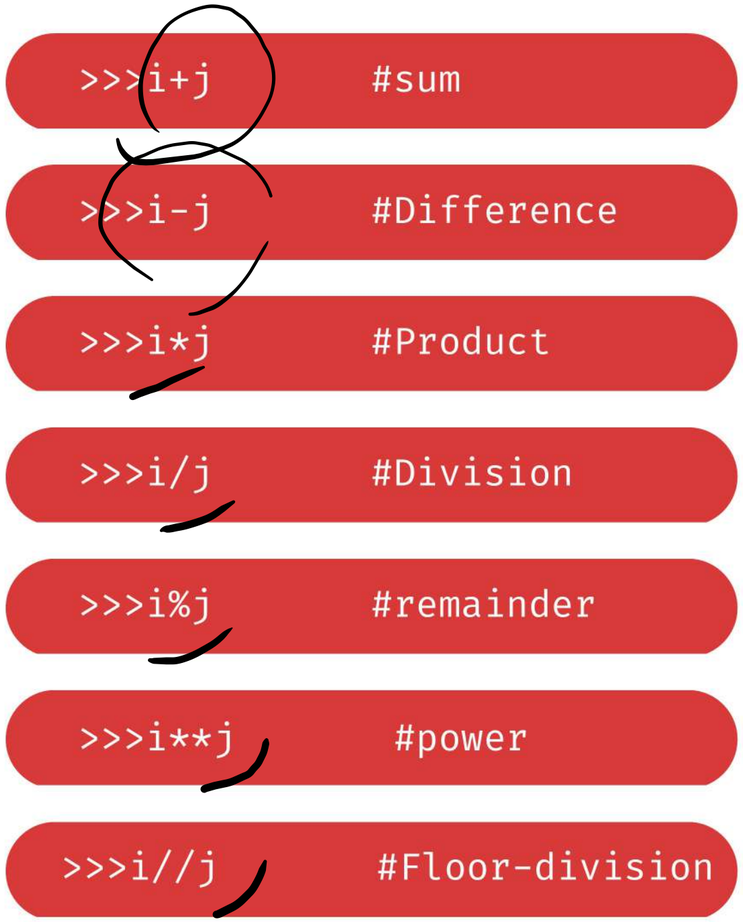
Operators

Operators

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

src: w3schools.com

Arithmetic Ops.



are used with numeric values to perform common mathematical operations

src: w3schools.com

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

src: w3schools.com

Comparison Ops.

```
>>> 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

True

False

src: w3schools.com

a, b → boolean.

not a → a: True → not a: False.

Logical Ops.

```
>>> 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
```

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

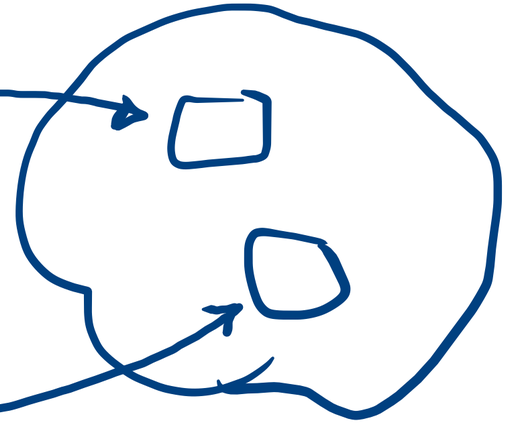
src: w3schools.com

== vs is

A is B → False

A = 12

B = 12



A == B → True

Tree = 30 → assignment

== vs = ---▷

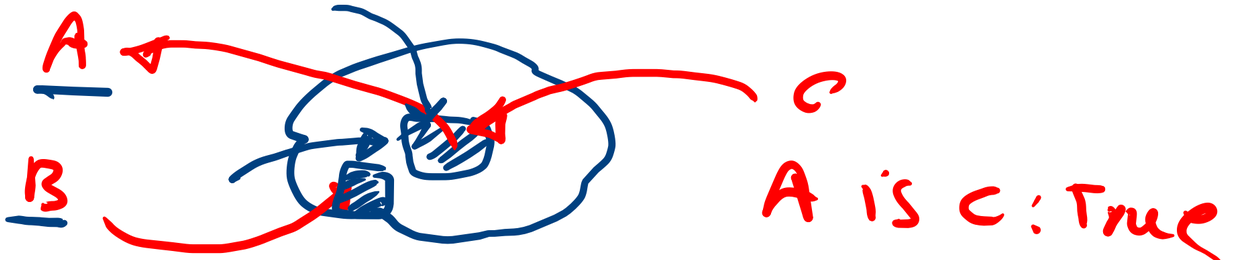
Comparison.

Tree == 30

check

A is B → False.

A is not B : True



A is not C : False

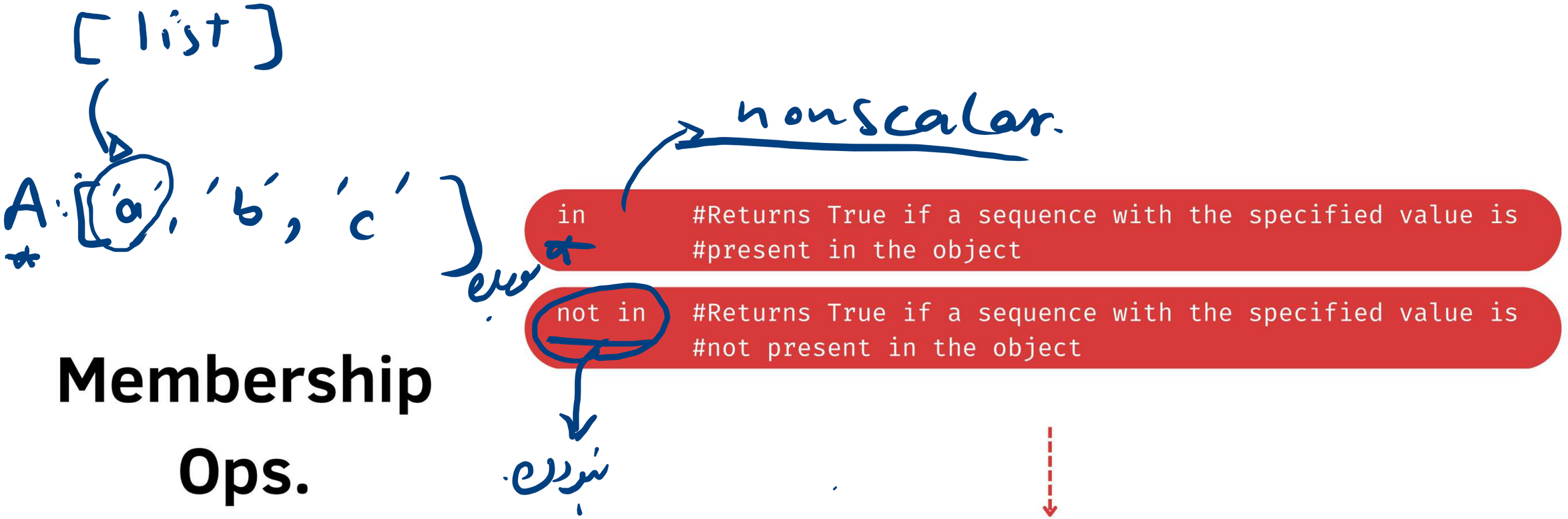
`is`	`#Returns True if both variables are the same object`
`is not`	`#Returns True if both variables are not the same object`

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.

src: w3schools.com



Membership Ops.

B = 'a'

B in A → True

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

src: w3schools.com

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
```

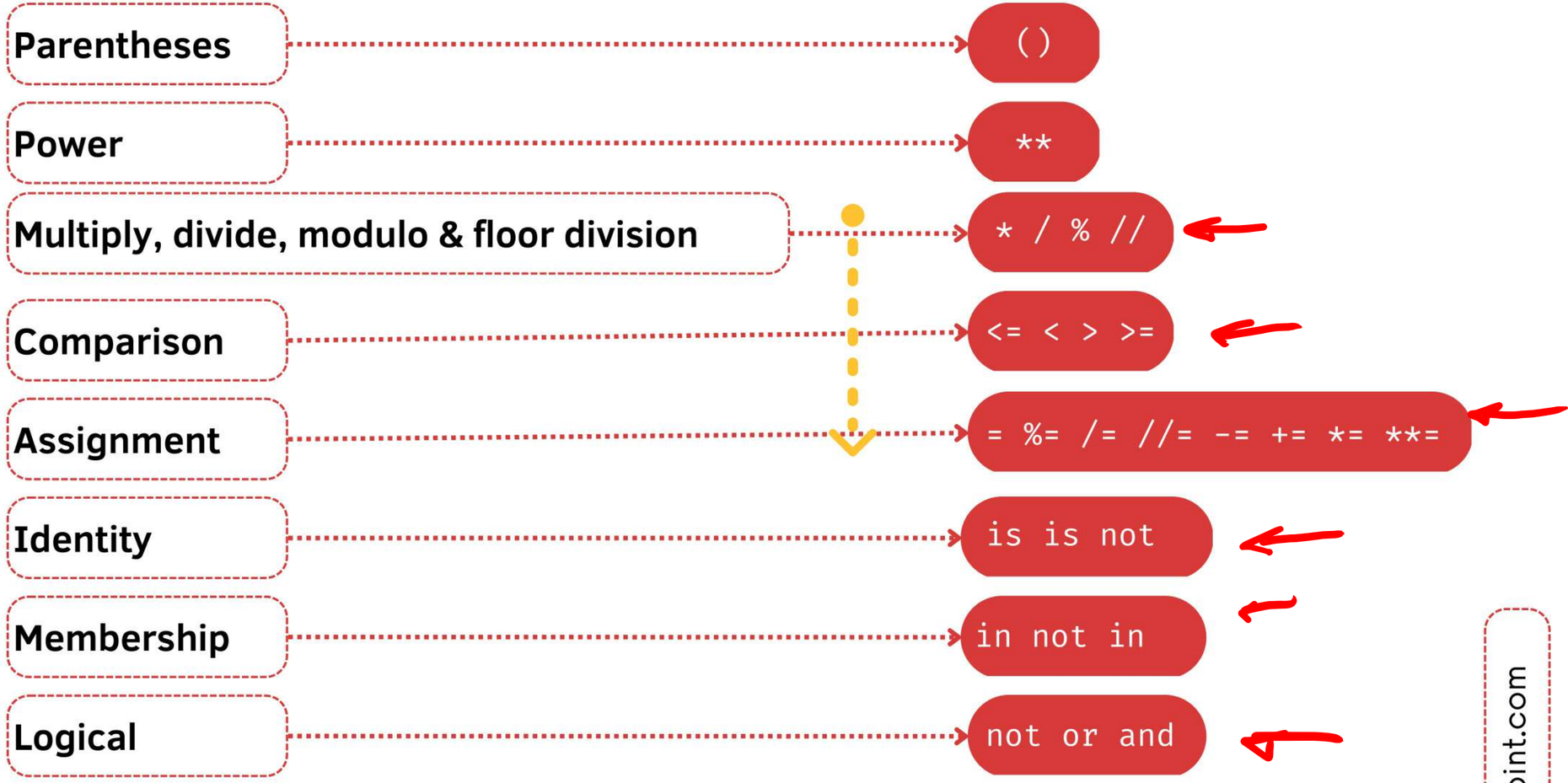
are used to compare (binary) numbers

src: w3schools.com

3 * 2 - 5 / 18

( )

Operator Precedence



src: tutorialspoint.com

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)
```

src: tutorialspoint.com

= 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 ?

Expressions

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

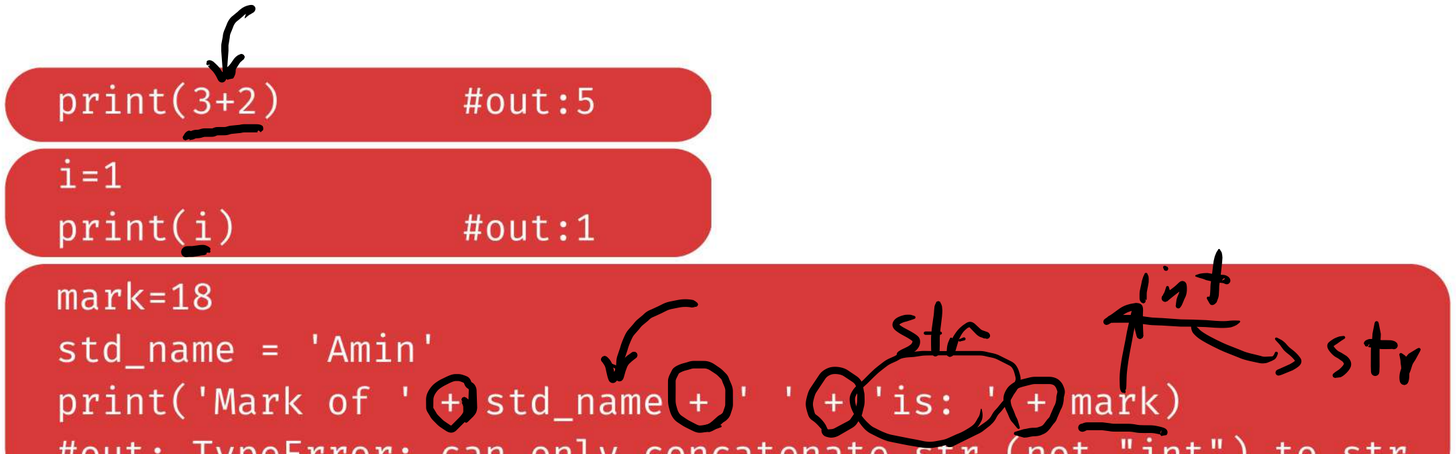
```
x = 25              # a statement
x = x + 10          # an expression
```

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
```

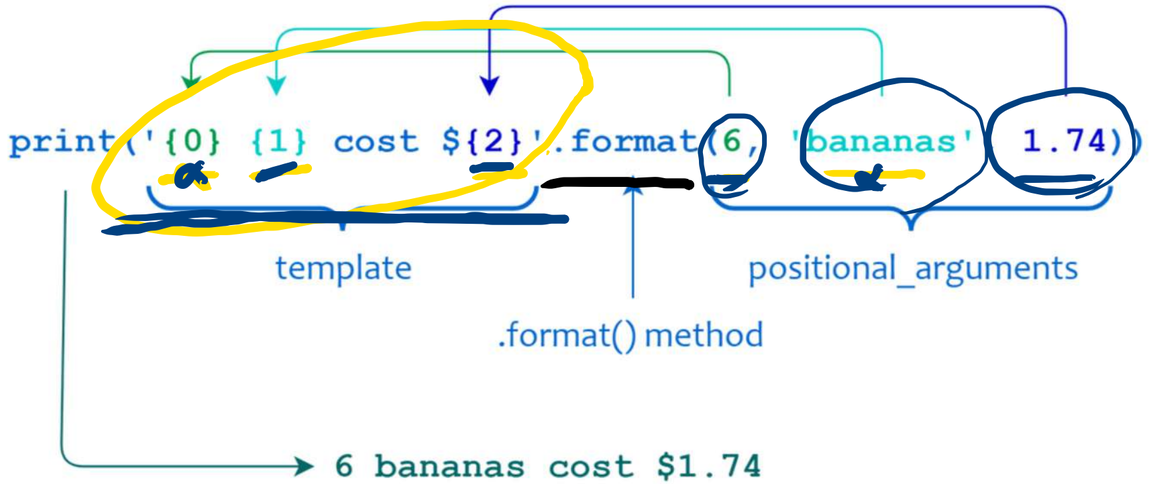



```
mark=18
std_name = 'Amin'
print('Mark of ', std_name, ' ', 'is: ', mark)
#out: Mark of  Amin   is:  18
```

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

```
mark=18
mark_str = str(mark)
std_name = 'Amin'
print('Mark of ' + std_name + ' ' + 'is: ' + mark_str)
#out: Mark of Amin is: 18
```

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

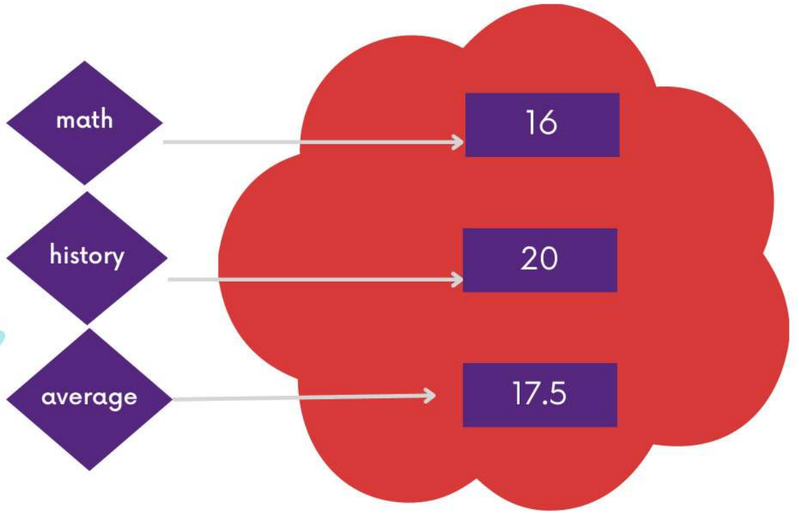
```
x = 25              # a statement
x = x + 10          # an expression
```

Abstracting Expressions

Abstracting Expressions

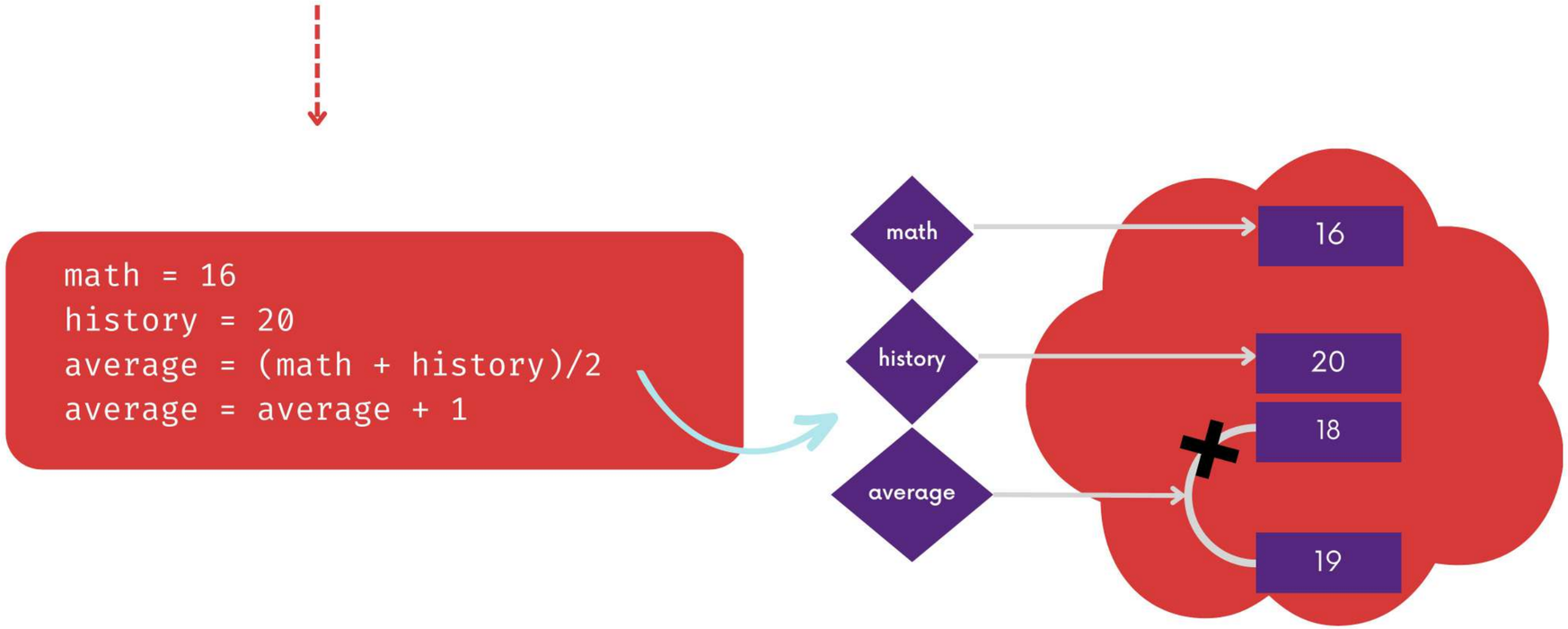
- to reuse names instead of values
- easier to change code later

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



Changing Binding

- 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.



Strings

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

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

- Concatenate Strings

```
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!

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