NHẬP MÔN LẬP TRÌNH
(Introduction to Programming)

Chapter 6 – "Pointer" with Python and Applications

Presenter: **Dr. Nguyen Dinh Long**

Email: dinhlonghcmut@gmail.com

Phone: +84 947 229 599

Google-site: https://sites.google.com/view/long-dinh-nguyen

12 Nov. 2022

Programming

·

Hamilton was a self-taught programmer, working in the US in the 1960's. Owing to the success of her previous work, Hamilton was the first programmer to be hired for the Apollo project. She became the Director of Software Engineering at the MIT Instrumentation lab. Her lab developed the on-board flight software for NASA's Apollo space project, which took humankind to the moon.

The achievement was a monumental task at a time when computer technology was in its infancy: The astronauts had access to only 72 kilobytes of computer memory (a 256-gigabyte cell phone today carries almost a million times more storage space). Programmers had to use paper punch cards to feed information into room-sized computers with no screen interface.

Margaret Hamilton, NASA's lead software engineer for the Apollo, stands next to the code she wrote by hand that took humanity to the moon in 1969.

Outline

- Overview of Computer, Computing and Programming
- Basic Concepts and Data Forms
- Variable and Function in Programming
- Sequence/Array and Its Structure
- Commands and Control Structure
- ● Pointer and Application
- Algorithm and Application Programming

References

Main:

- Maurizio Gabbrielli and Simone Martini, 2010. *Programming Languages: Principles and Paradigms*, Springer.
- Cao Hoàng Trụ, 2004. *Ngôn ngữ lập trình- Các nguyên lý và mô hình*, Nhà xuất bản Đại học Quốc gia Tp. Hồ Chí Minh

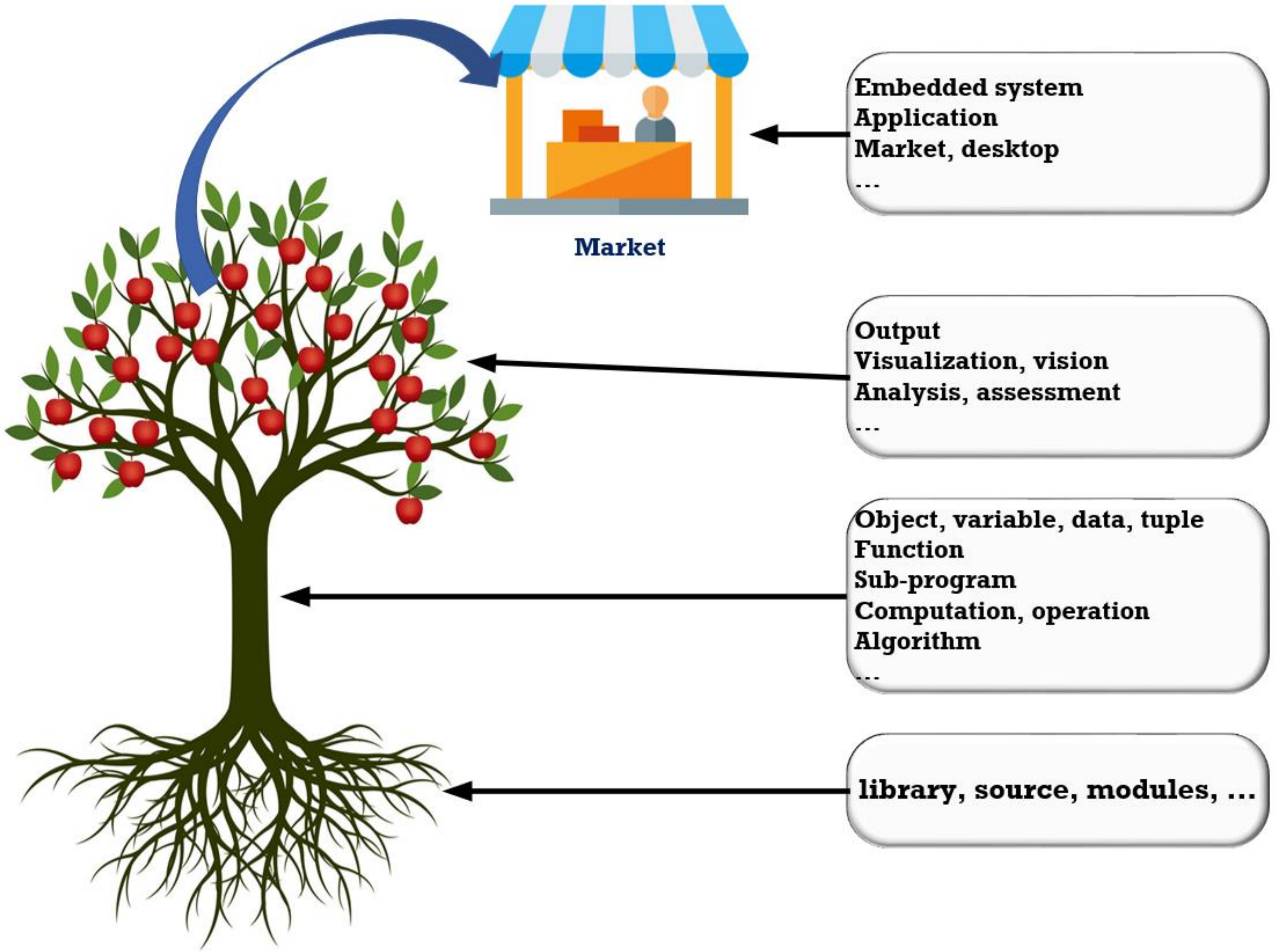
More:

- Wes McKinney, 2013. *Python for Data Analysis*, O'Reilly Media.
- Guido van Rossum, Fred L. Drake, Jr.,, 2012. *The Python Library Reference*, Release 3.2.3.

- Slides here are collected and modified from several sources in Universities and Internet.

Computer programs

- General structure:

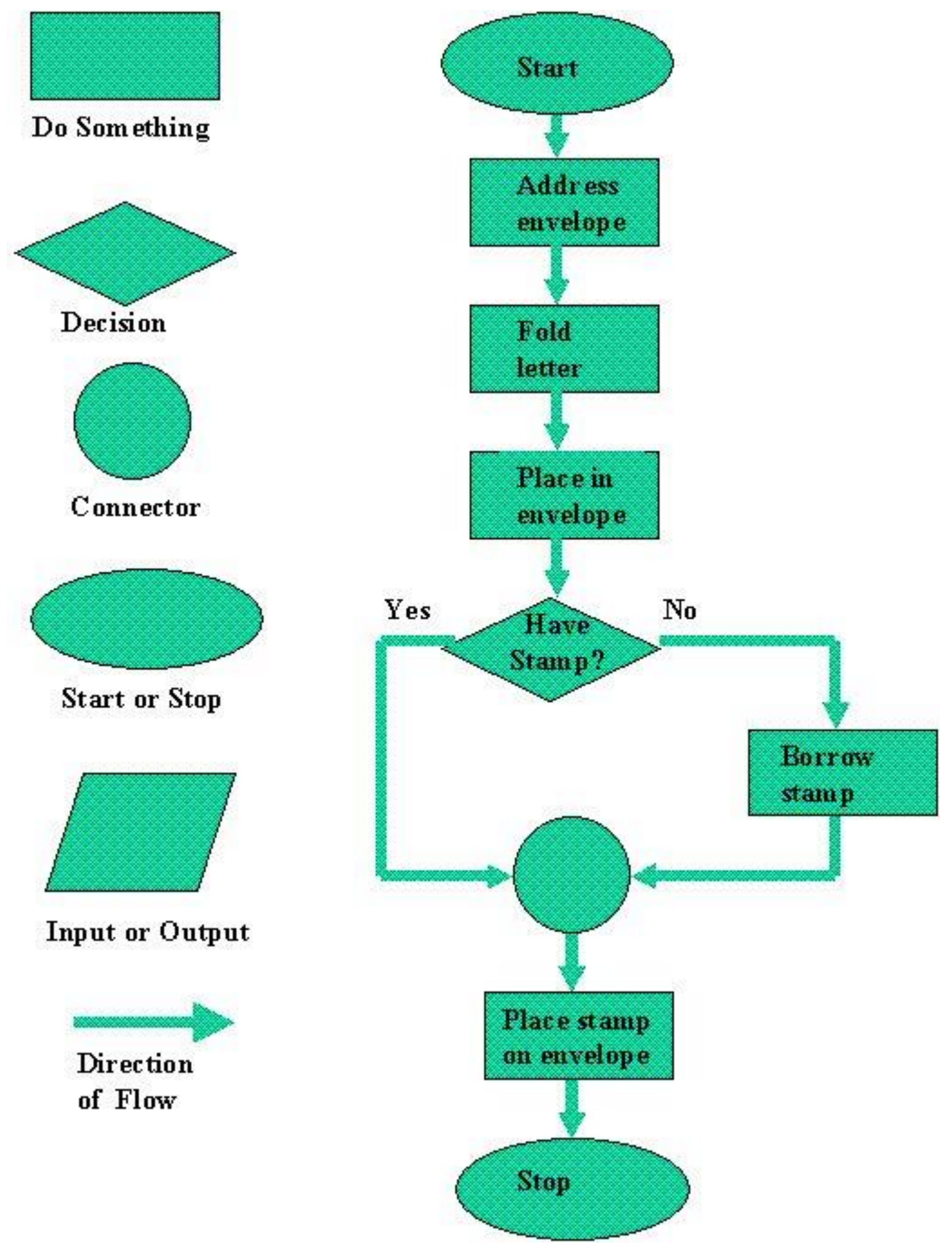


Content of Chapter 6

1. What are the Pointers?
2. Pointers are used in PYTHON?
3. Uses of the Pointer in Python
4. Multiple Pointers Approach

Structure of Computer programs

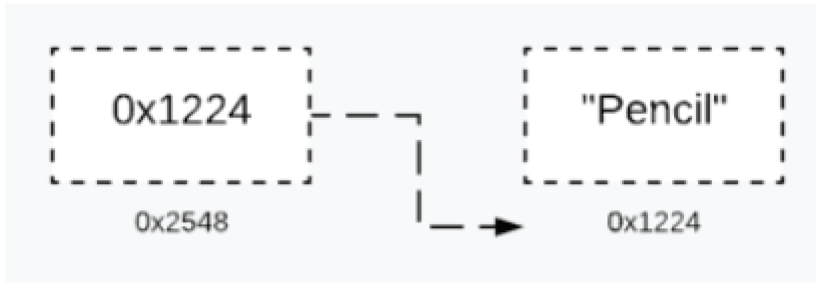
- **Computer programming:**
  - Objects
  - Types
  - Variables
  - Methods
  - Sequences/Arrays



Pointer and Applications

❑ What are the Pointers?

Pointers in the context of high-level languages like C++ or Java are nothing but variables that stores the memory address of other variables or objects.



We can reference another variable stored in a different memory location using a pointer in high-level programming languages.

Unlike variables, pointers deal with address rather than values.

Pointer and Applications

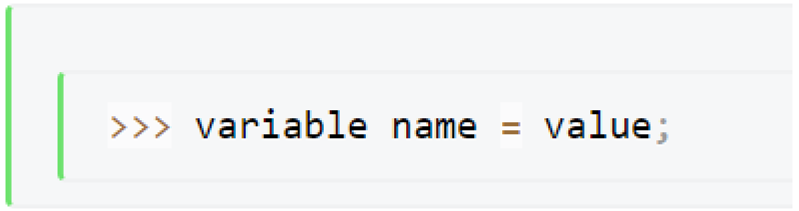
❑ Pointers are used in PYTHON?

Python doesn’t have pointers like other languages for explicit use but is implemented under the hood.

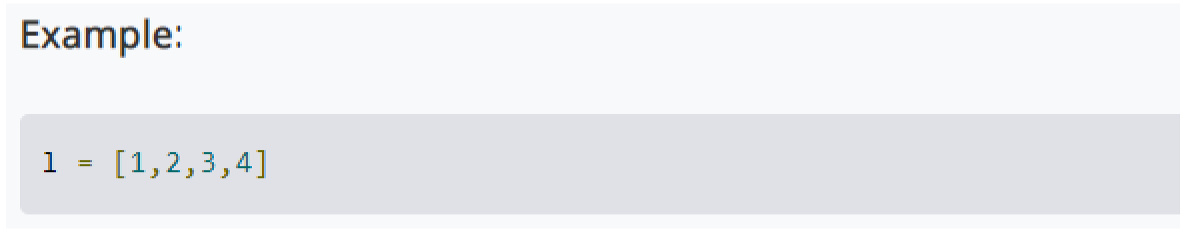
Types such as list, dictionary, class, and objects, example, in Python behave like pointers under the hood.

The assignment operator = in Python automatically creates and assigns a pointer to the variable.

Syntax of Pointer in Python

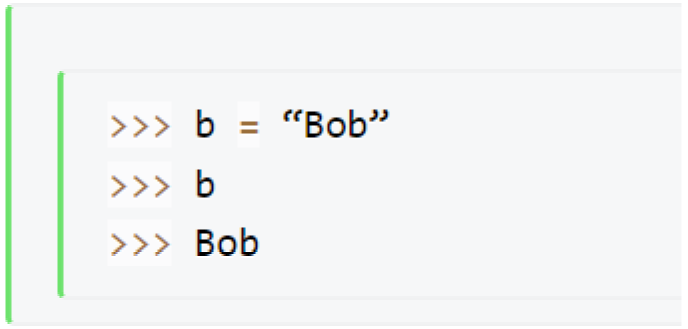


```
>>> variable name = value;
```



Example:

```
l = [1,2,3,4]
```



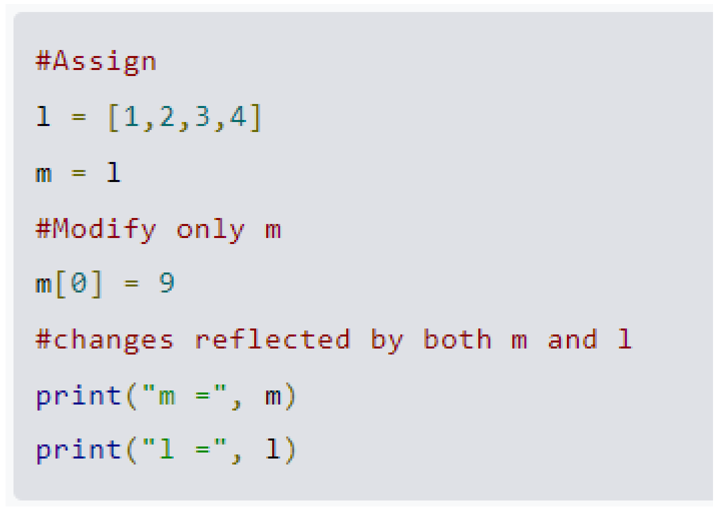
```
>>> b = “Bob”
>>> b
>>> Bob
```

Pointer and Applications

❑ Pointers are used in PYTHON?

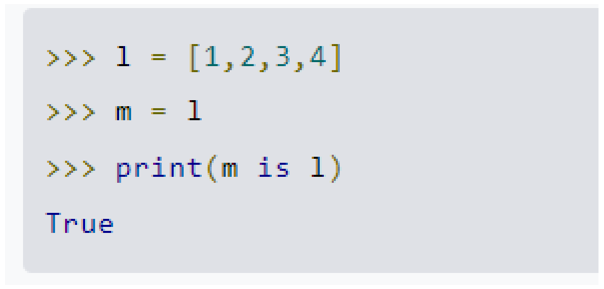
Python doesn’t have pointers like other languages for explicit use but is implemented under the hood.

If you assign the same list object both to two different variables m and l, they will point to the same memory location.



```
#Assign
l = [1,2,3,4]
m = l
#Modify only m
m[0] = 9
#changes reflected by both m and l
print("m =", m)
print("l =", l)
```

To make sure that “m” and “l” are pointing to same object lets use is operator.



```
>>> l = [1,2,3,4]
>>> m = l
>>> print(m is l)
True
```

- Also, it is important to note that python deletes an object only when all the pointers to the object is deleted.
- Use *del m* or *del l* will only delete the variable not the underlying object to which it was referencing.
- Python will still keep the object in the memory until all the pointers pointing to the list object are deleted.

Pointer and Applications

- **Pointers are used in number and strings?**

The answer is **NO**. Simples types such as int, str, float, bool, … do not behave like a pointer.

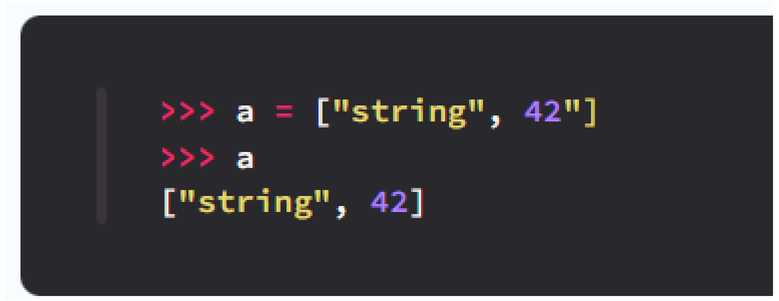
On a final note, we conclude that Python doesn’t have pointers but the behavior of pointers is implemented under the hood by the python for a few types and objects.

Pointer and Applications

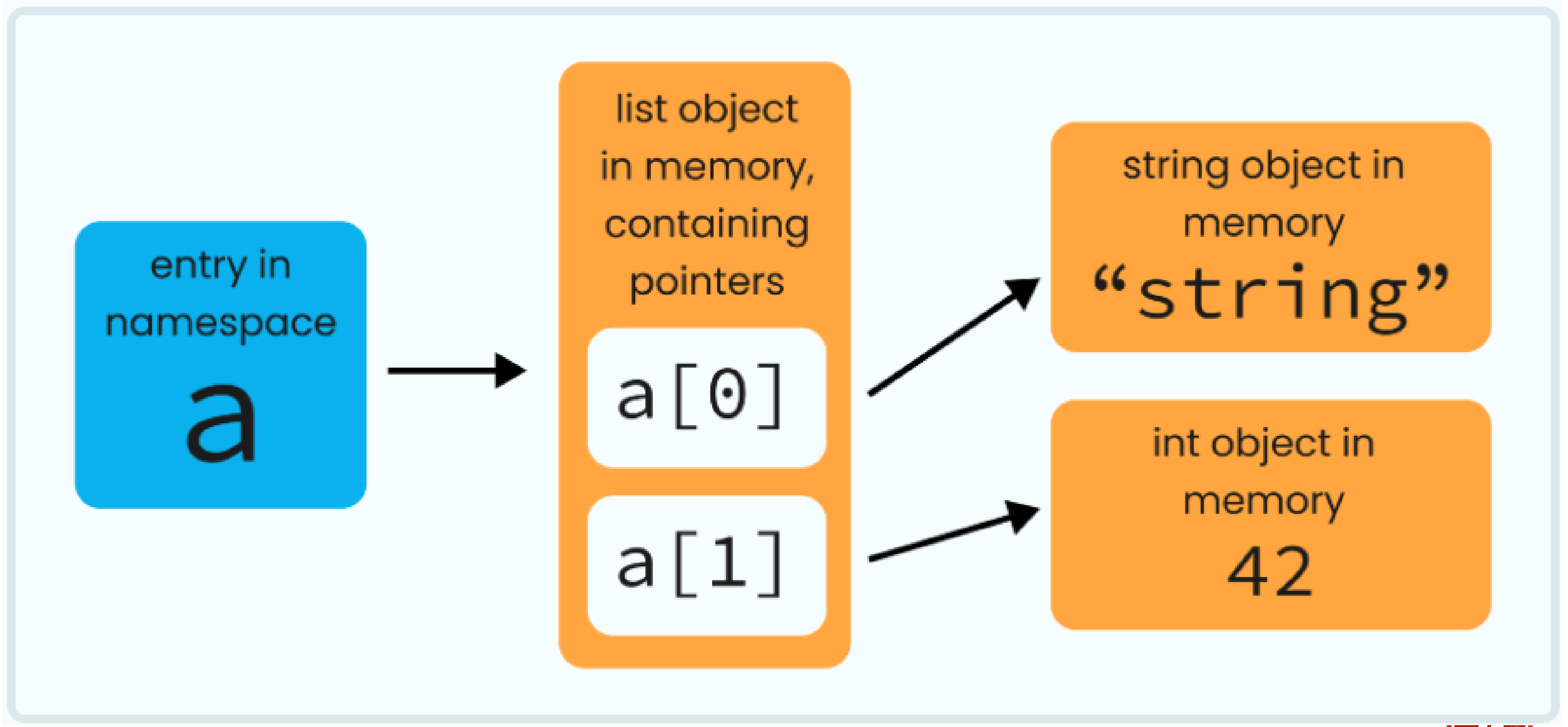
❑ **Pointers are used in PYTHON?**

Python doesn't have pointers like other languages for explicit use but is implemented under the hood.

As an example, let's consider a list object with the name my_list and two arbitrary elements.



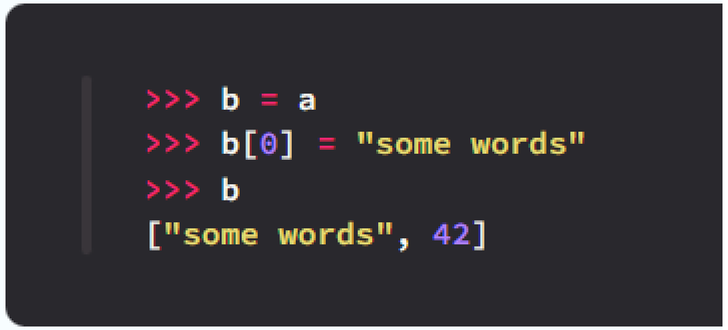
```
>>> a = ["string", 42"]
>>> a
["string", 42]
```



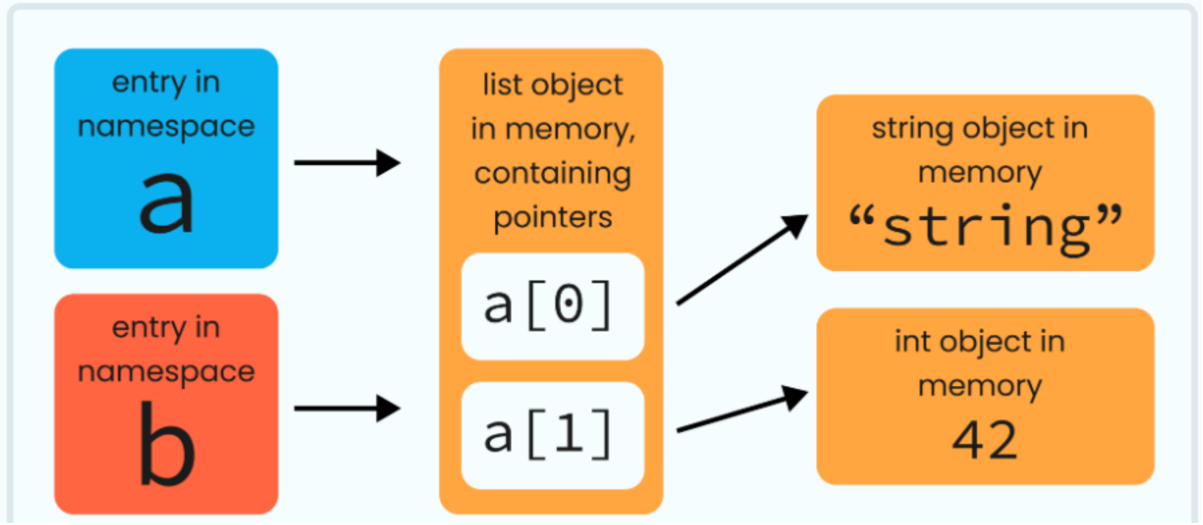
Pointer and Applications

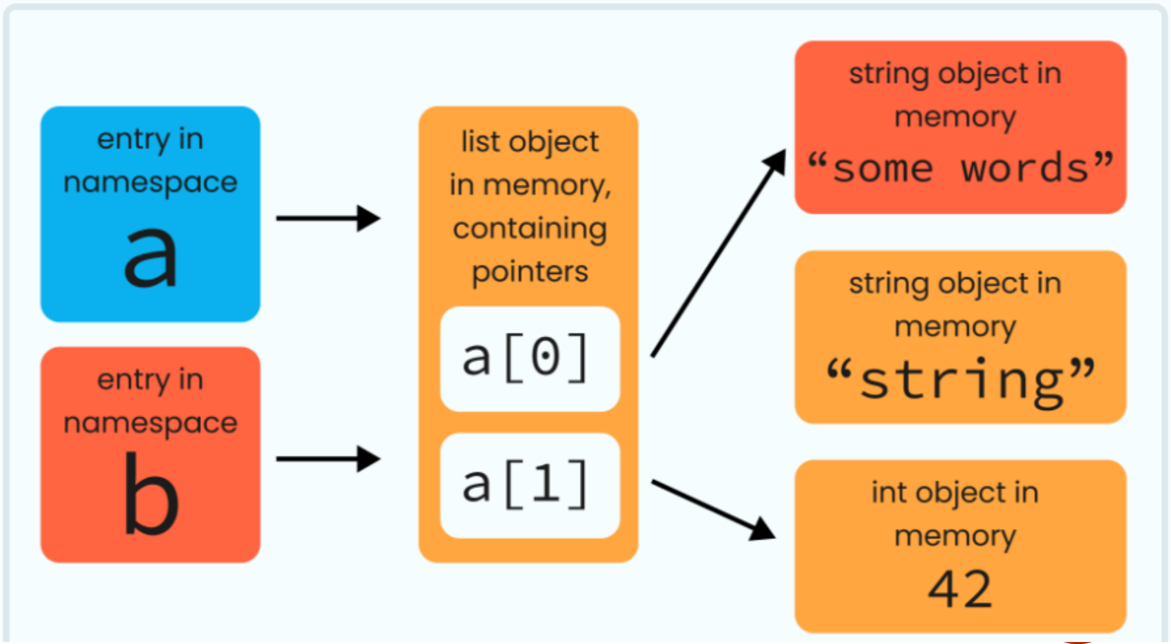
❑ Pointer aliasing in Python

What’s happened above is that, in the line where we set b = a, we didn’t actually make a new list object.



```
>>> b = a
>>> b[0] = "some words"
>>> b
["some words", 42]
```

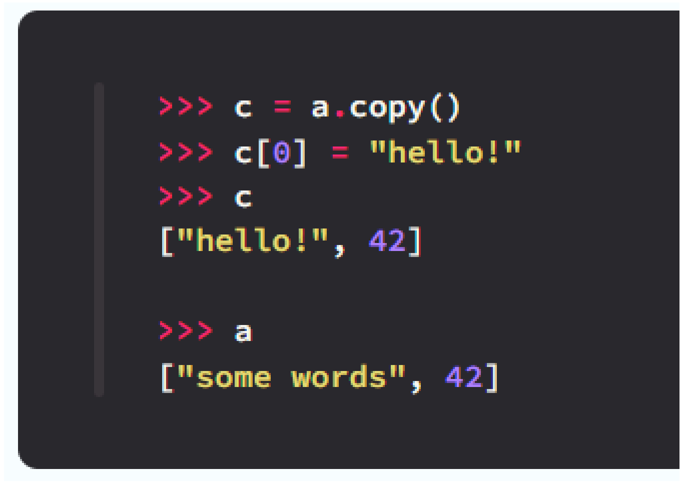

entry in namespace a
entry in namespace b
list object in memory, containing pointers
a[0]
a[1]
string object in memory “string”
int object in memory 42


entry in namespace a
entry in namespace b
list object in memory, containing pointers
a[0]
a[1]
string object in memory “some words”
string object in memory “string”
int object in memory 42

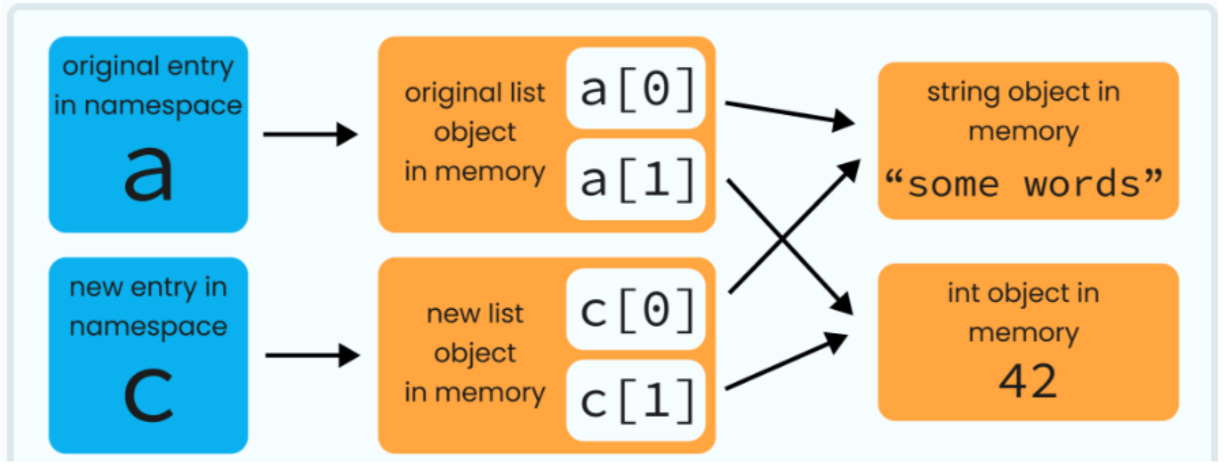
Pointer and Applications

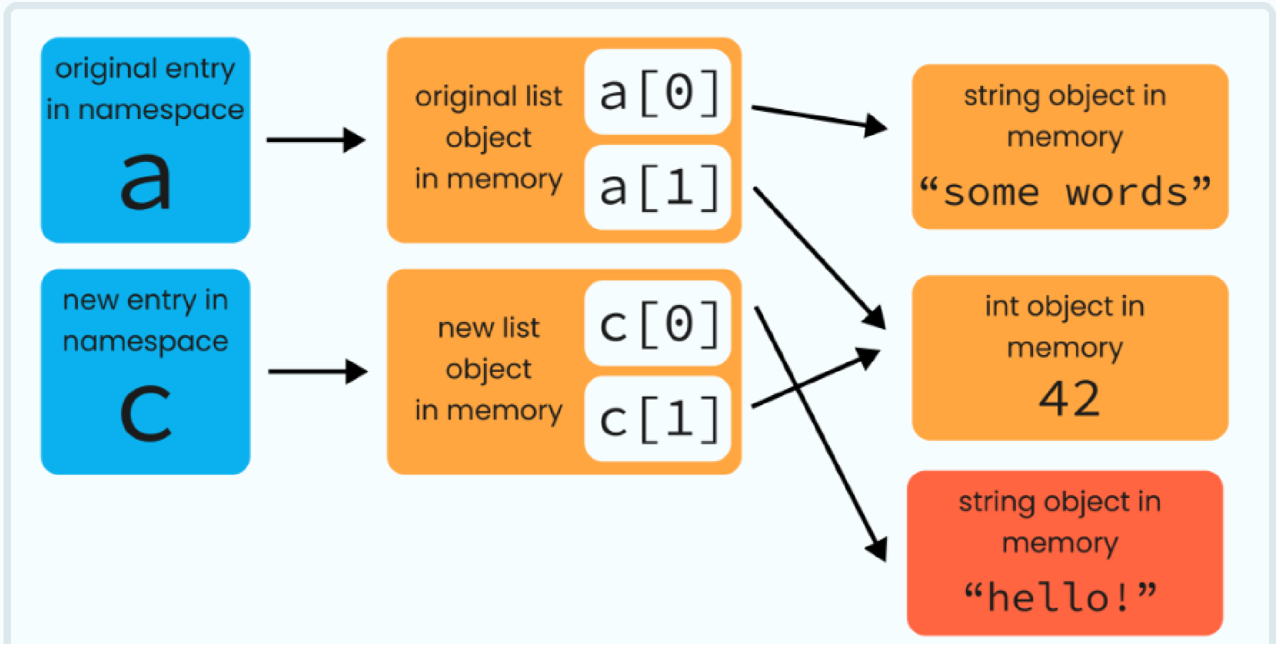
❑ Pointer aliasing in Python

We can use the copy method on our original list object, and this does create a new list object.



```
>>> c = a.copy()
>>> c[0] = "hello!"
>>> c
["hello!", 42]

>>> a
["some words", 42]
```





Pointer and Applications

❑ Can Create Pointers in Python?

An example of creating pointers with an isinstance () function to prove that it is an object type.

We will see all possible datatypes in Python with isinstance() function; this way, you will learn how to declare all datatypes in python as well.

```
// assigning an integer value
a = 2
print(a)
// checking if integer is an object or not
print(isinstance(a, object))
```

```
// assigning a string value
b = "Bob"
print(b)
// checking if string is an object or not
print(isinstance(b, object))
```

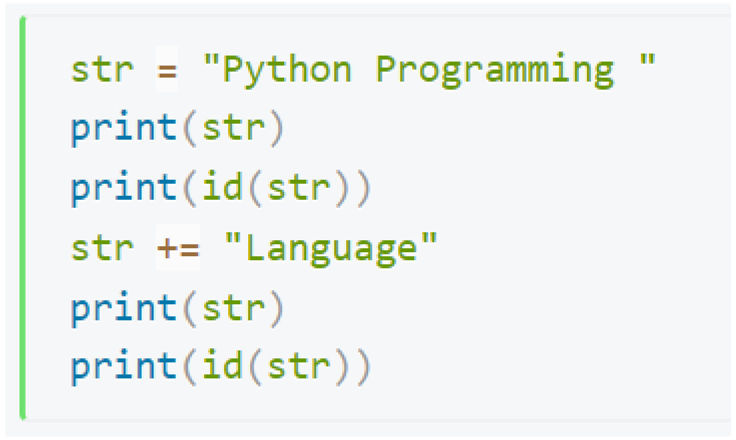
```
// assigning a list value
inputList = [1,2,3]
print(inputList)
// checking if list is an object or not
print(isinstance(inputList, object))
```

```
//assigning a set value
inputSet = {10,20,30}
print(inputSet)
// checking if set is an object or not
print(isinstance(inputSet, object))
```

Pointer and Applications

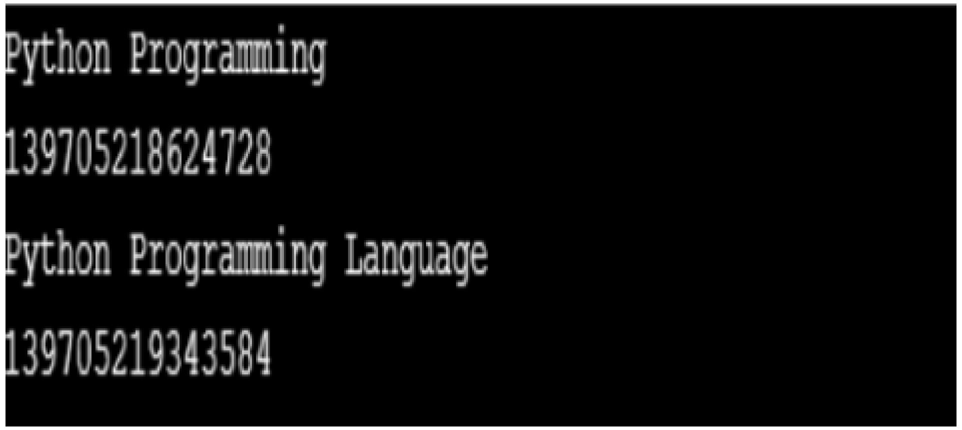
❑ Can Create Pointers in Python?

On an immutable object like a String, we can do an appending as mentioned below:



```
str = "Python Programming "
print(str)
print(id(str))
str += "Language"
print(str)
print(id(str))
```

Output:



```
Python Programming
139705218624728
Python Programming Language
139705219343584
```

Pointer and Applications

❑ Uses of the Pointer in Python

Pointers are used in C and C++ widely. With Pointers, dynamic memory allocation is possible. Pointers can be declared as variables holding the memory address of another variable.

- Pointers Arithmetic Operations: Pointers have four arithmetic operators
  - Increment Operator : ++
  - Decrement Operator: —
  - Addition Operator : +
  - Subtraction Operator : –

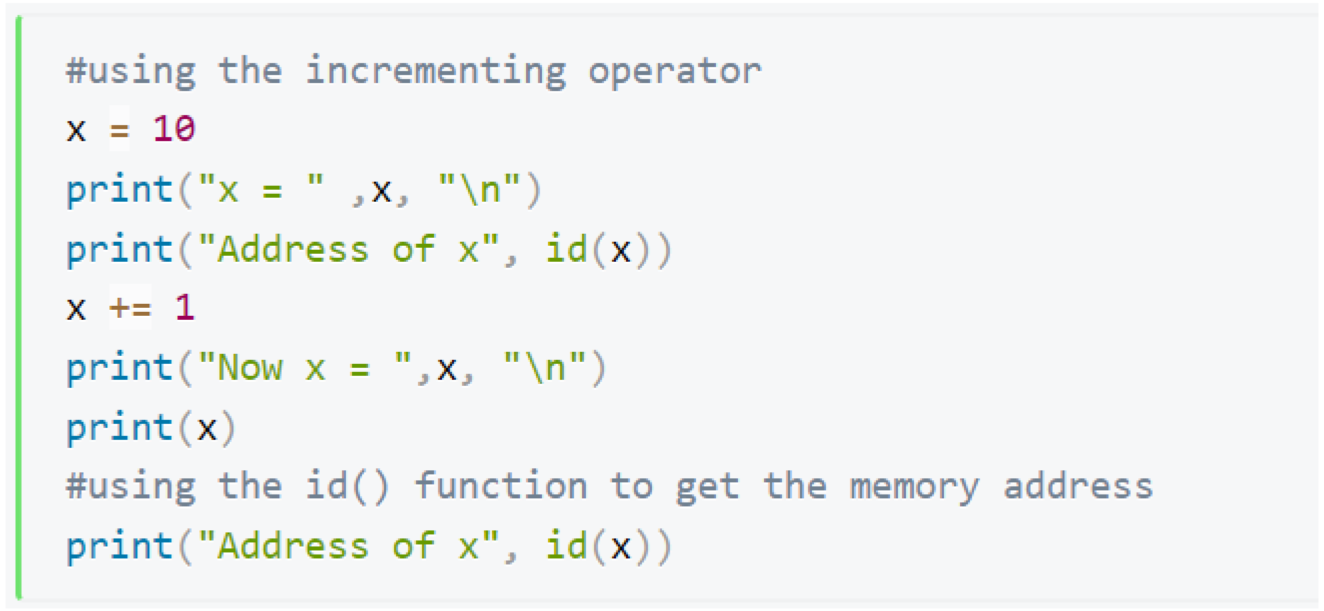
Arithmetic operations are performed with the use of arithmetic operators.

In the below programs, we have used *id() function*, which returns the object’s memory address.

Pointer and Applications

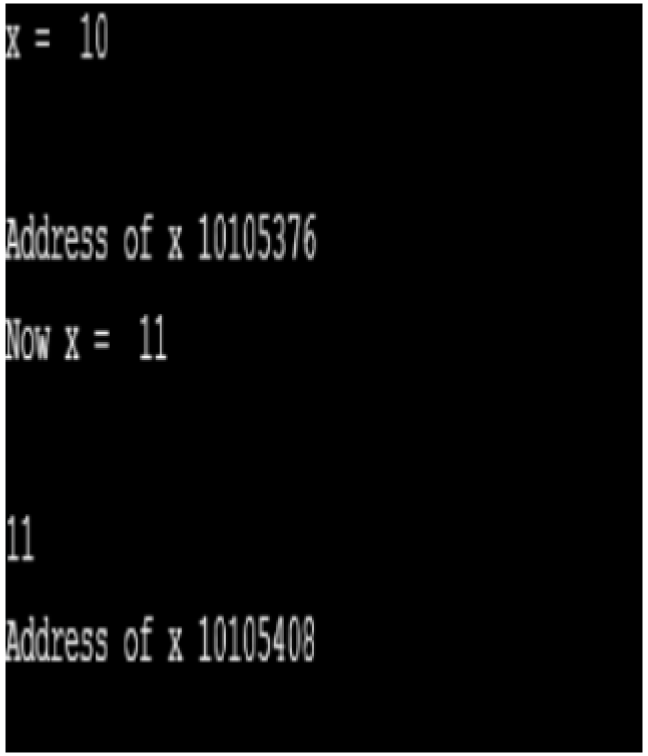
❑ Uses of the Pointer in Python

Increment operator: It increments the value by 1



```
#using the incrementing operator
x = 10
print("x = " ,x, "\n")
print("Address of x", id(x))
x += 1
print("Now x = ",x, "\n")
print(x)
#using the id() function to get the memory address
print("Address of x", id(x))
```

Output:

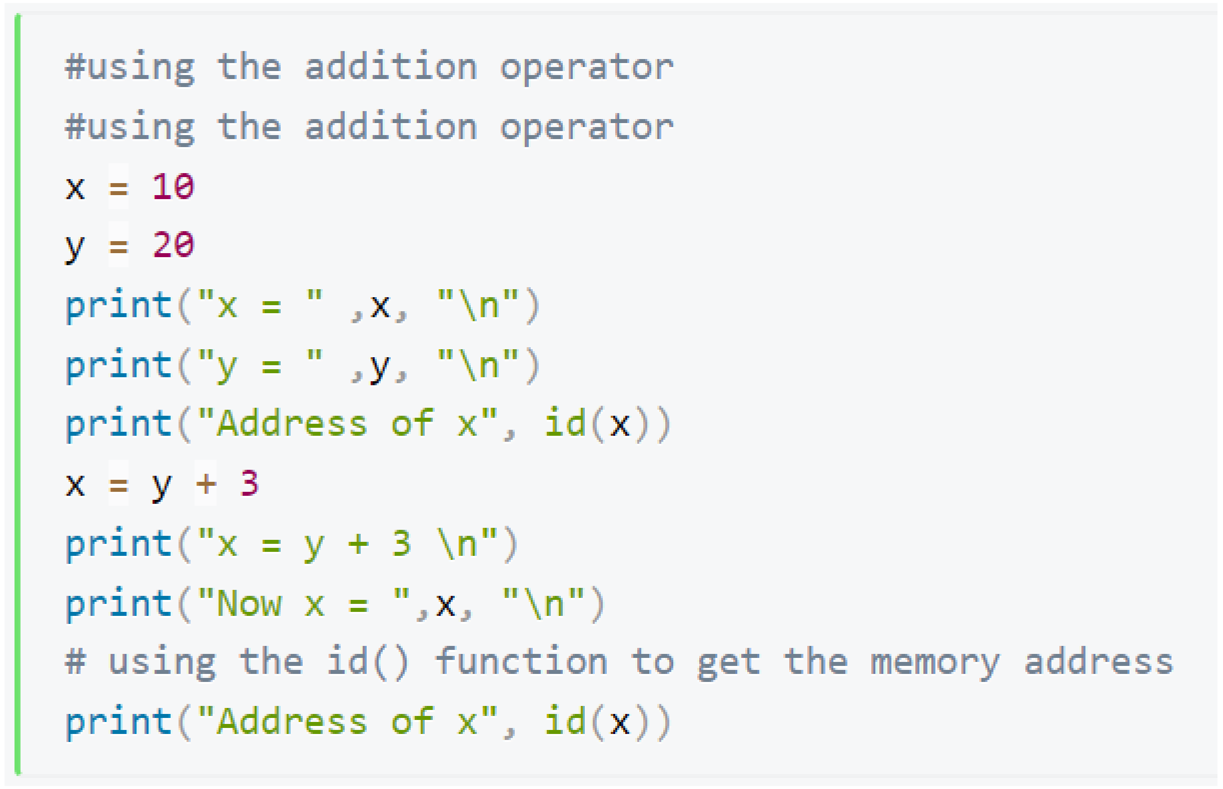


```
x =  10


Address of x 10105376
Now x =  11


11
Address of x 10105408
```

Pointer and Applications

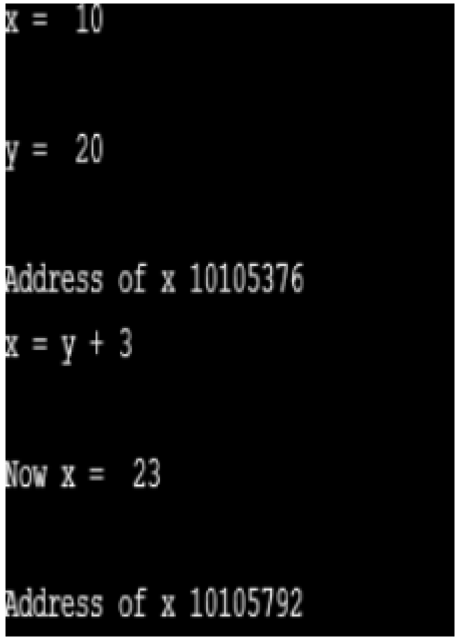
❑ Uses of the Pointer in Python

Addition Operator: It performs the addition of two operands



```
#using the addition operator
#using the addition operator
x = 10
y = 20
print("x = " ,x, "\n")
print("y = " ,y, "\n")
print("Address of x", id(x))
x = y + 3
print("x = y + 3 \n")
print("Now x = ",x, "\n")
# using the id() function to get the memory address
print("Address of x", id(x))
```

Output:

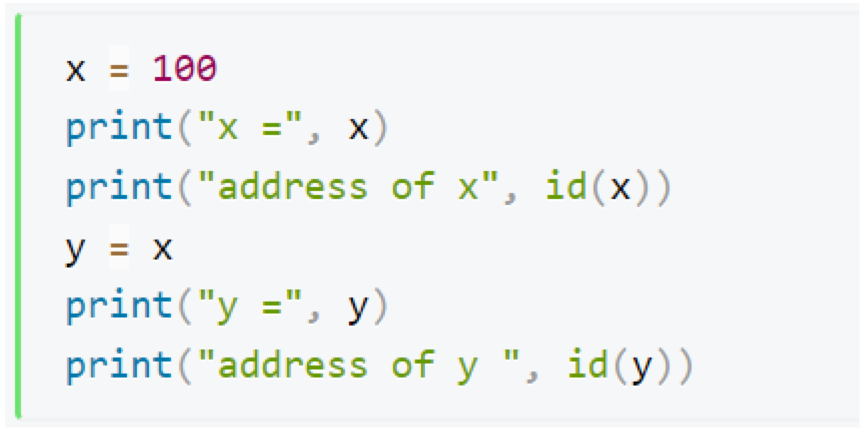


```
x =  10


y =  20


Address of x 10105376
x = y + 3


Now x =  23


Address of x 10105792
```

Pointer and Applications

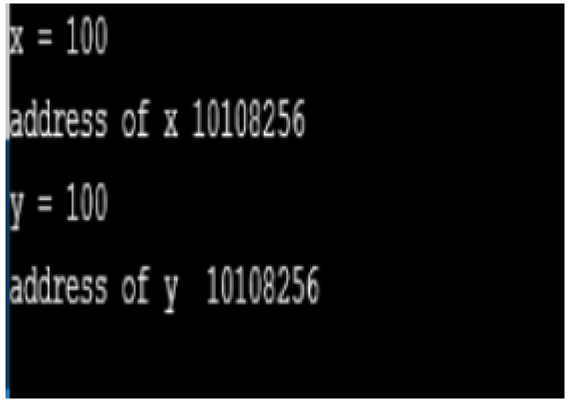
❑ Uses of the Pointer in Python

In this example, we are declaring two variables, x and y, where y is equal to x, which now points to the same memory address as that of x.



```
x = 100
print("x =", x)
print("address of x", id(x))
y = x
print("y =", y)
print("address of y ", id(y))
```

Output:

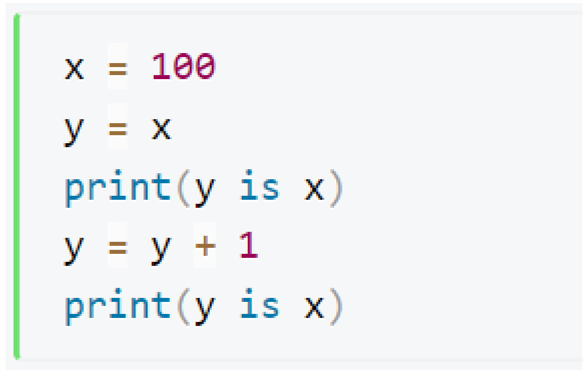


```
x = 100

address of x 10108256

y = 100

address of y  10108256
```

Pointer and Applications

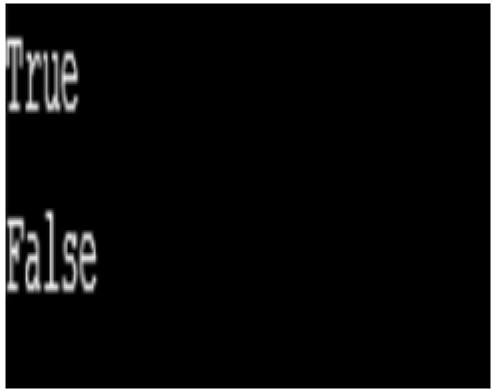
❑ Uses of the Pointer in Python

In this example, we are declaring two variables x and y, where y is equal to x, which is true, but when we increment the value of y by one, the output turns to be false.



```python
x = 100
y = x
print(y is x)
y = y + 1
print(y is x)
```

Output:

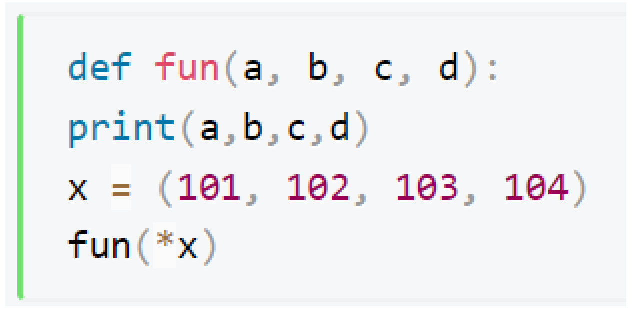


```
True
False
```

Pointer and Applications

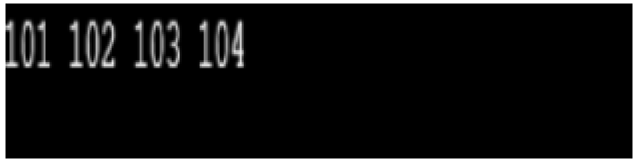
❑ Pointers to Pointers in Python

See the Example 1:



```
def fun(a, b, c, d):
print(a,b,c,d)
x = (101, 102, 103, 104)
fun(*x)
```

Output:

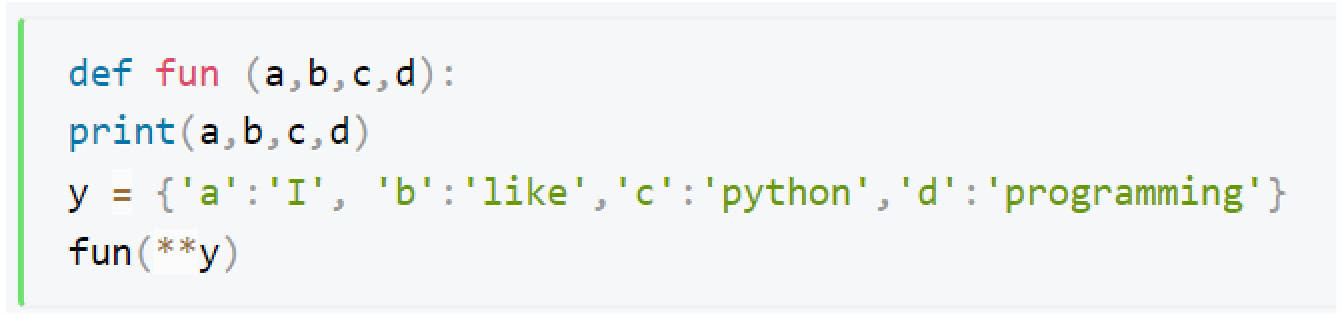


```
101 102 103 104
```

Pointer and Applications

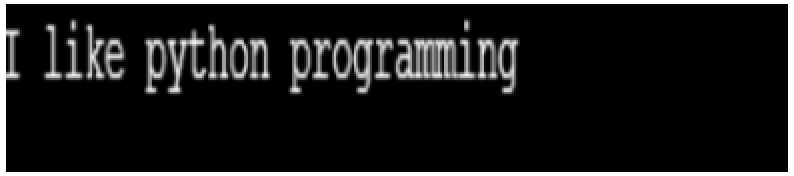
❑ Pointers to Pointers in Python

See the Example 2:



```python
def fun (a,b,c,d):
print(a,b,c,d)
y = {'a':'I', 'b':'like','c':'python','d':'programming'}
fun(**y)
```

Output:

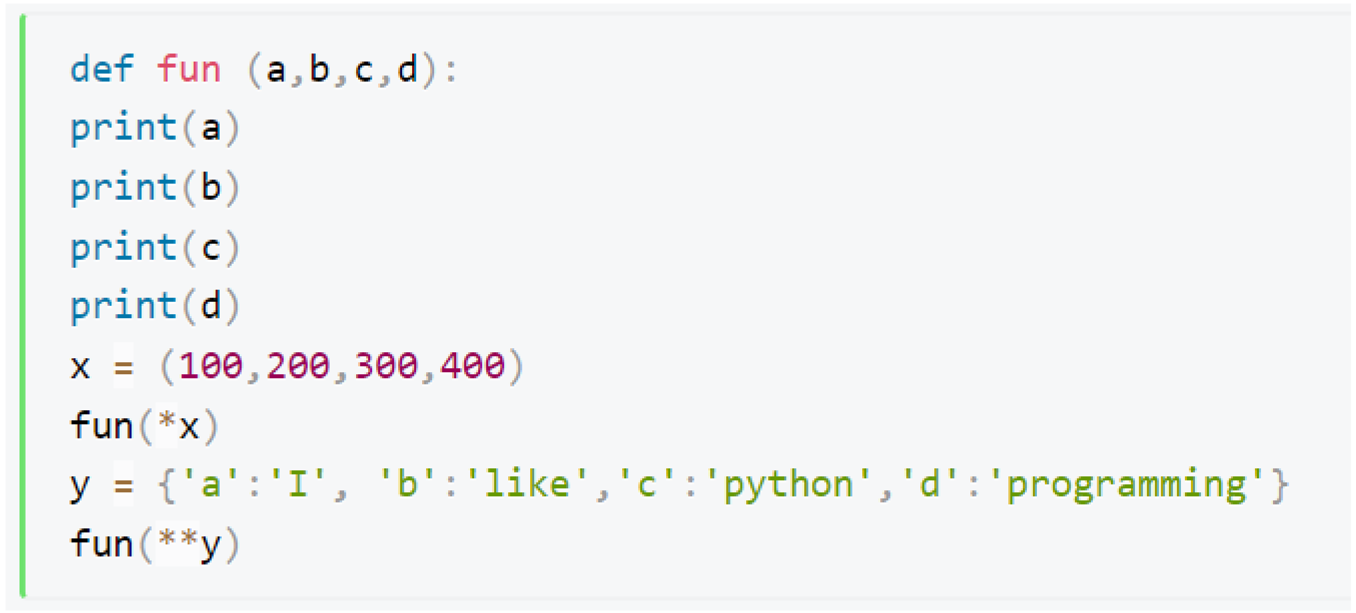


```
I like python programming
```

Pointer and Applications

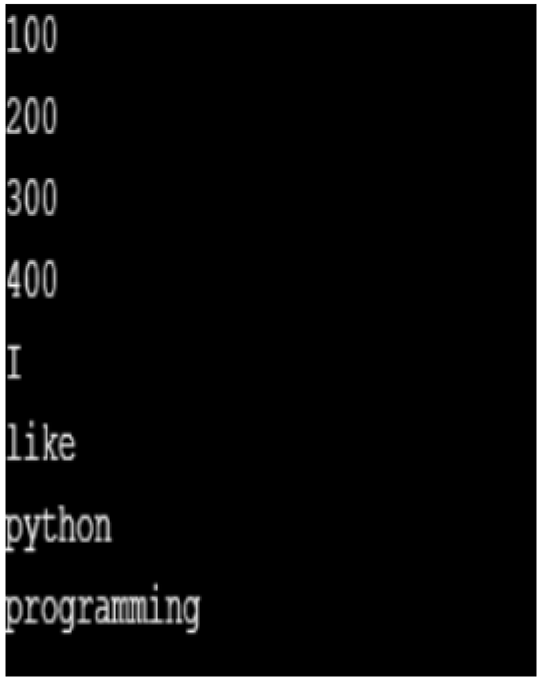
❑ Pointers to Pointers in Python

Putting Example 1 and Example 2 together:



```
def fun (a,b,c,d):
print(a)
print(b)
print(c)
print(d)
x = (100,200,300,400)
fun(*x)
y = {'a':'I', 'b':'like','c':'python','d':'programming'}
fun(**y)
```

Output:



```
100
200
300
400
I
like
python
programming
```

Pointer and Applications

Applications

Pointer and Applications

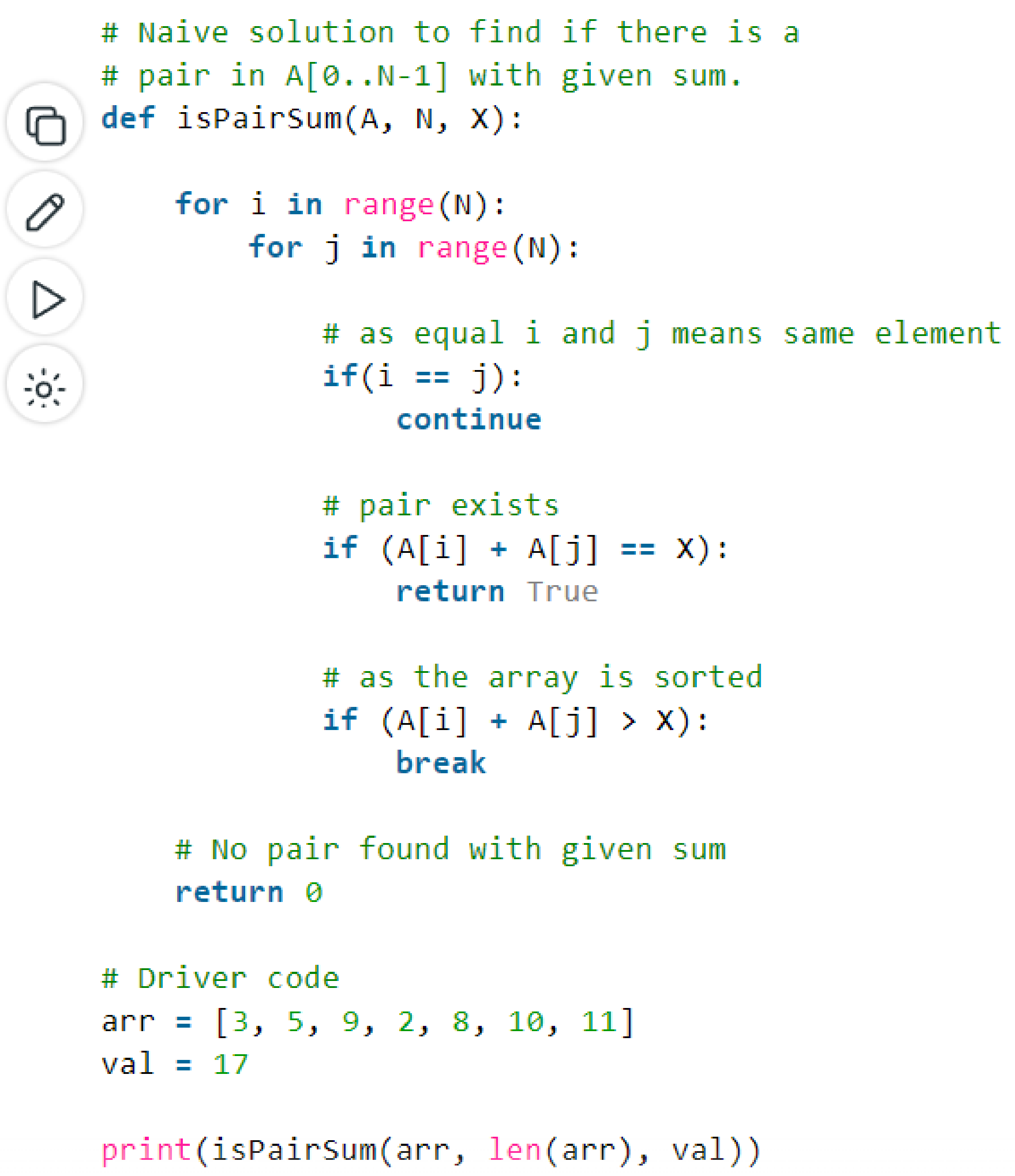
❑ Program for Two Pointers Technique:

Two pointers is really an easy and effective technique which is typically used for searching pairs in a sorted array.

Given a sorted array A (sorted in ascending order), having N integers, find if there exists any pair of elements

(A[i], A[j]) such that their sum is equal to X.

Let’s see the naive solution.



```python
# Naive solution to find if there is a
# pair in A[0..N-1] with given sum.
def isPairSum(A, N, X):

    for i in range(N):
        for j in range(N):

            # as equal i and j means same element
            if(i == j):
                continue

            # pair exists
            if (A[i] + A[j] == X):
                return True

            # as the array is sorted
            if (A[i] + A[j] > X):
                break

    # No pair found with given sum
    return 0

# Driver code
arr = [3, 5, 9, 2, 8, 10, 11]
val = 17

print(isPairSum(arr, len(arr), val))
```

Pointer and Applications

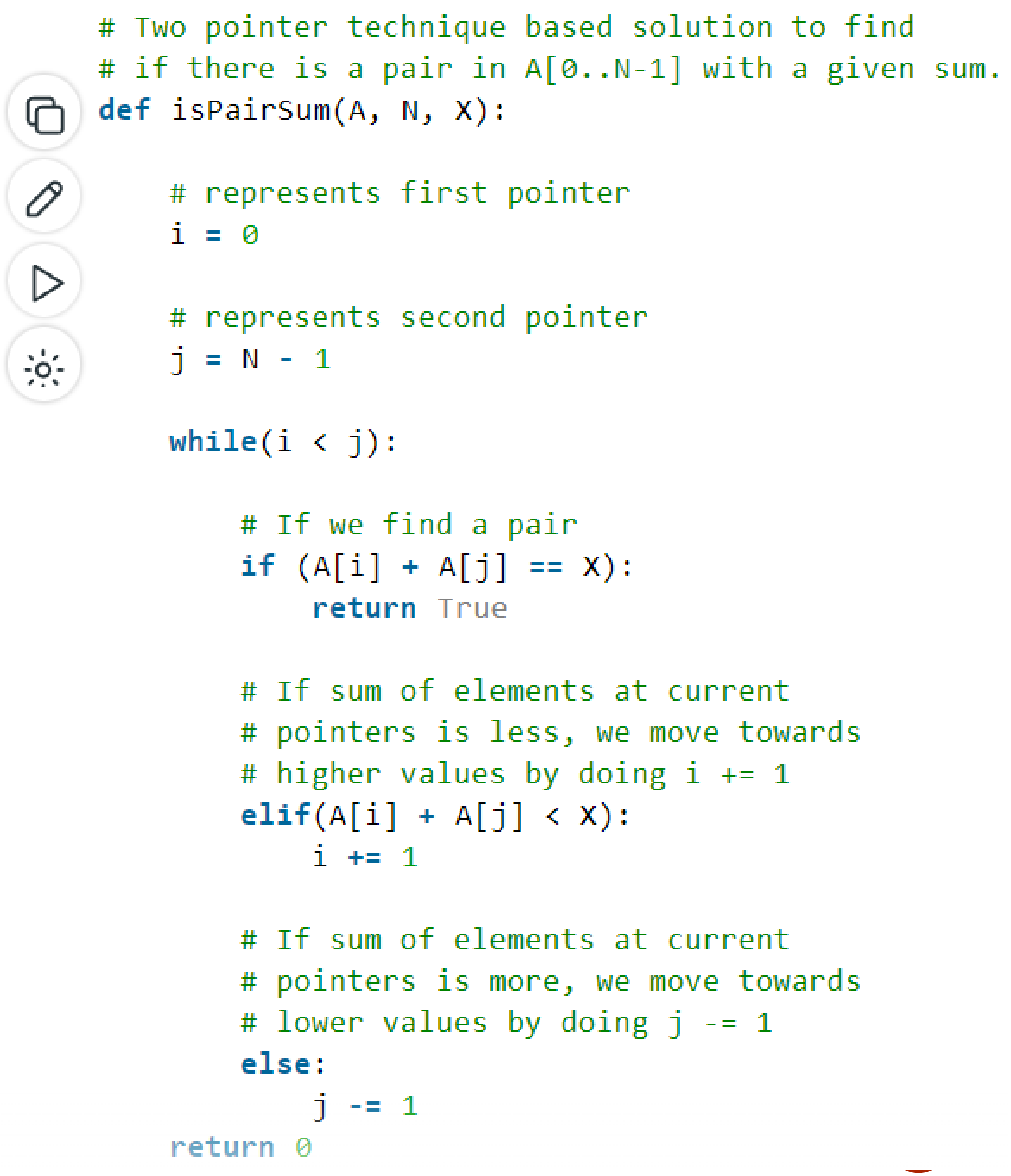
❑ Program for Two Pointers Technique:

Now let’s see how the two-pointer technique works.

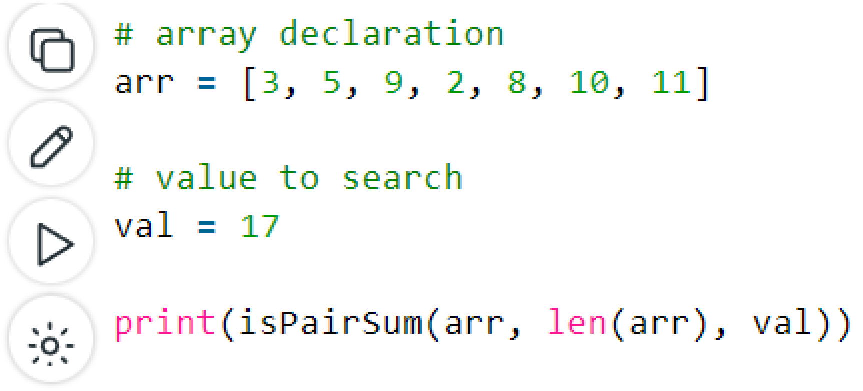
We take two pointers, one representing the first element and other representing the last element of the array, and then we add the values kept at both the pointers.

If their sum is smaller than X then we shift the left pointer to right or if their sum is greater than X then we shift the right pointer to left, in order to get closer to the sum.

We keep moving the pointers until we get the sum as X.



```python
# Two pointer technique based solution to find
# if there is a pair in A[0..N-1] with a given sum.
def isPairSum(A, N, X):

    # represents first pointer
    i = 0

    # represents second pointer
    j = N - 1

    while(i < j):

        # If we find a pair
        if (A[i] + A[j] == X):
            return True

        # If sum of elements at current
        # pointers is less, we move towards
        # higher values by doing i += 1
        elif(A[i] + A[j] < X):
            i += 1

        # If sum of elements at current
        # pointers is more, we move towards
        # lower values by doing j -= 1
        else:
            j -= 1
    return 0
```



```python
# array declaration
arr = [3, 5, 9, 2, 8, 10, 11]

# value to search
val = 17

print(isPairSum(arr, len(arr), val))
```

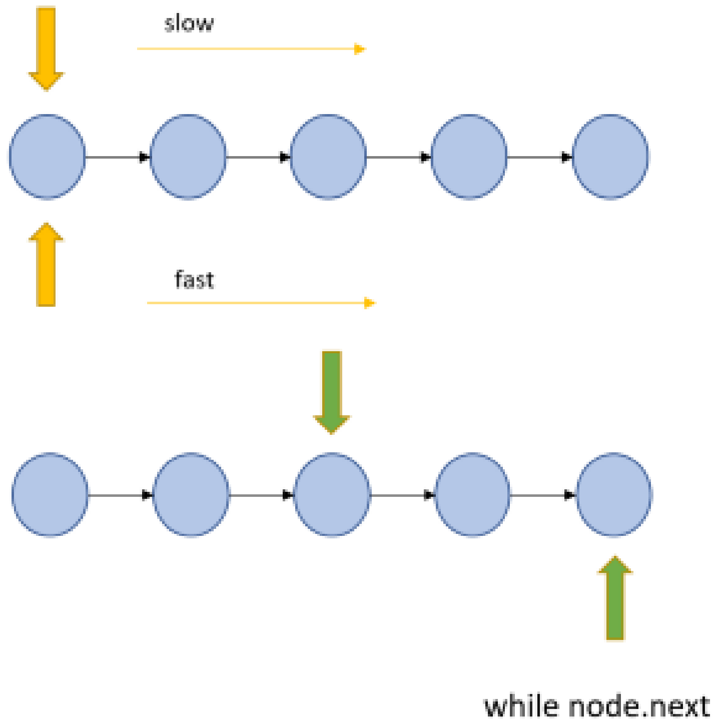
Pointer and Applications

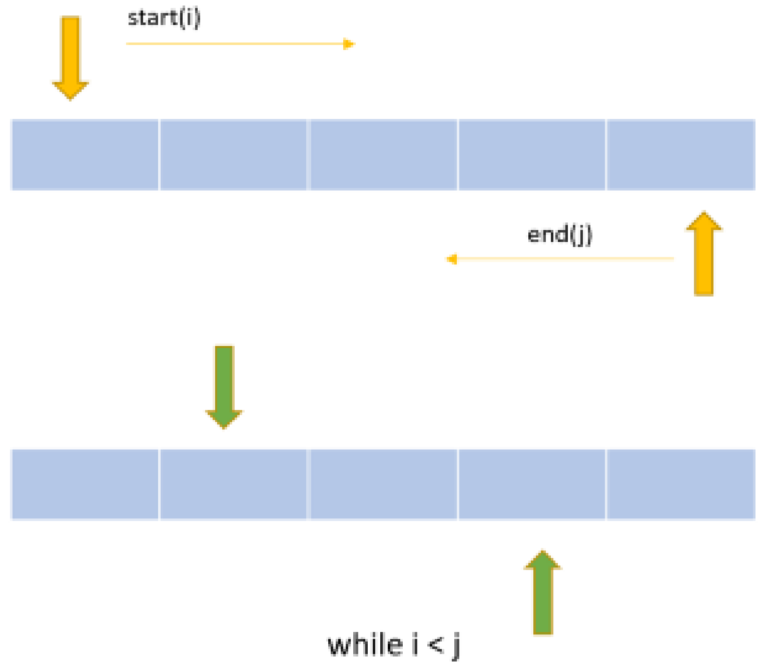
❑ Two Pointers Approach:

Two pointer algorithm is one of the most commonly asked questions in any programming interview.

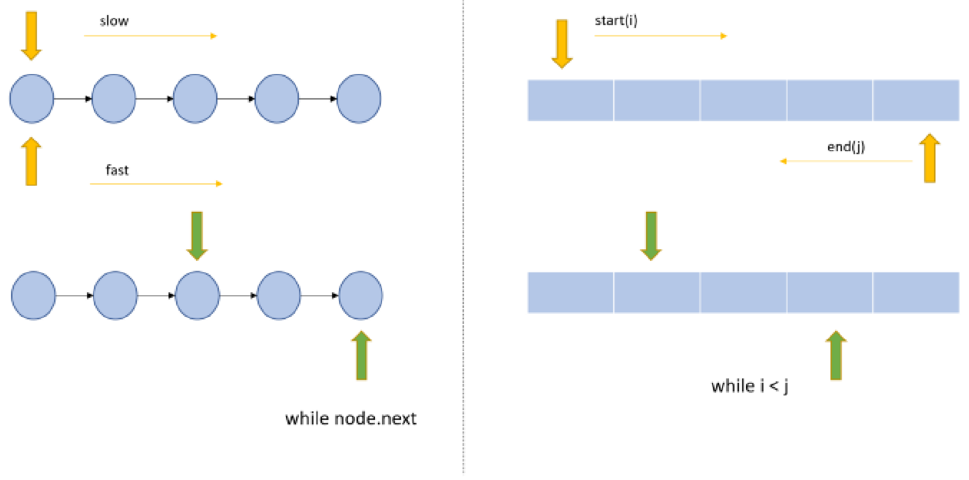
This approach optimizes the runtime by utilizing some order (not necessarily sorting) of the data. It is generally applied on lists (arrays) and linked lists. This is generally used to search pairs in a sorted array.

This approach works in constant space.





Pointer and Applications



❑ Two Pointers Approach:

As in the above pic, the two-pointer approach has three main steps:

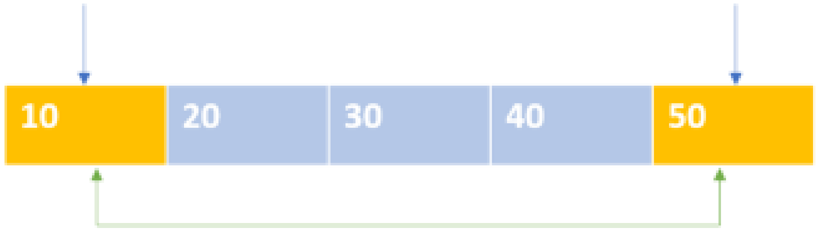
- Pointer Initialization — Starting points. Pointers can be at any place depending upon what we are trying to achieve.
- Pointer movement — This will decide how we converge towards the solution.
- Stop condition — This decides when do we stop.

Pointer and Applications

- **Two Pointers Approach:**

Example: **Reverse an array in place**

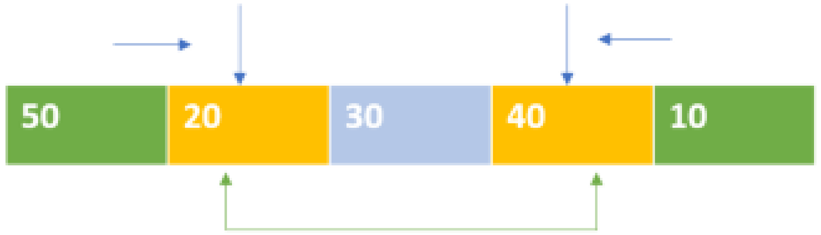
```
def reverseArray(array):
    start, end = 0, len(array)-1
    while start< end:
        array[start], array[end] = array[end] , array[start]
        start += 1
        end -= 1

array = [10, 20, 30, 40, 50]
reverseArray(array)
print(array)
```



start = 0
end = 4

swap elements at these positions

Move start towards right (start += 1)
Move end towards left (end -= 1)



start = 1
end = 3

swap elements at these positions

Move start towards right (start += 1)
Move end towards left (end -= 1)
At this we reach our break condition as start is no longer smaller than end (they become same 2)

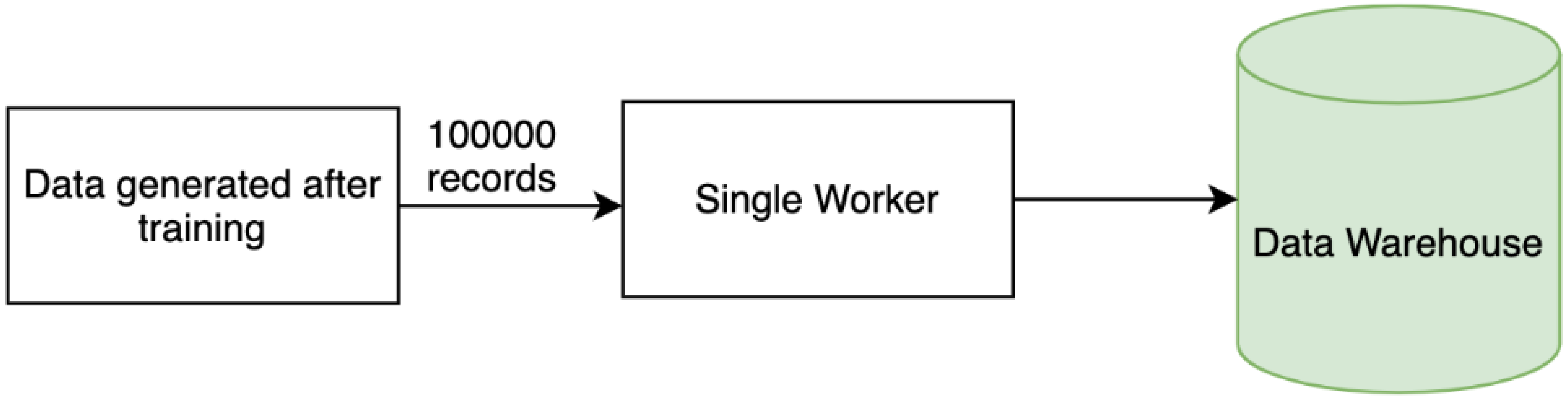
Parallel Processing

The story of Globalization

Logistic management

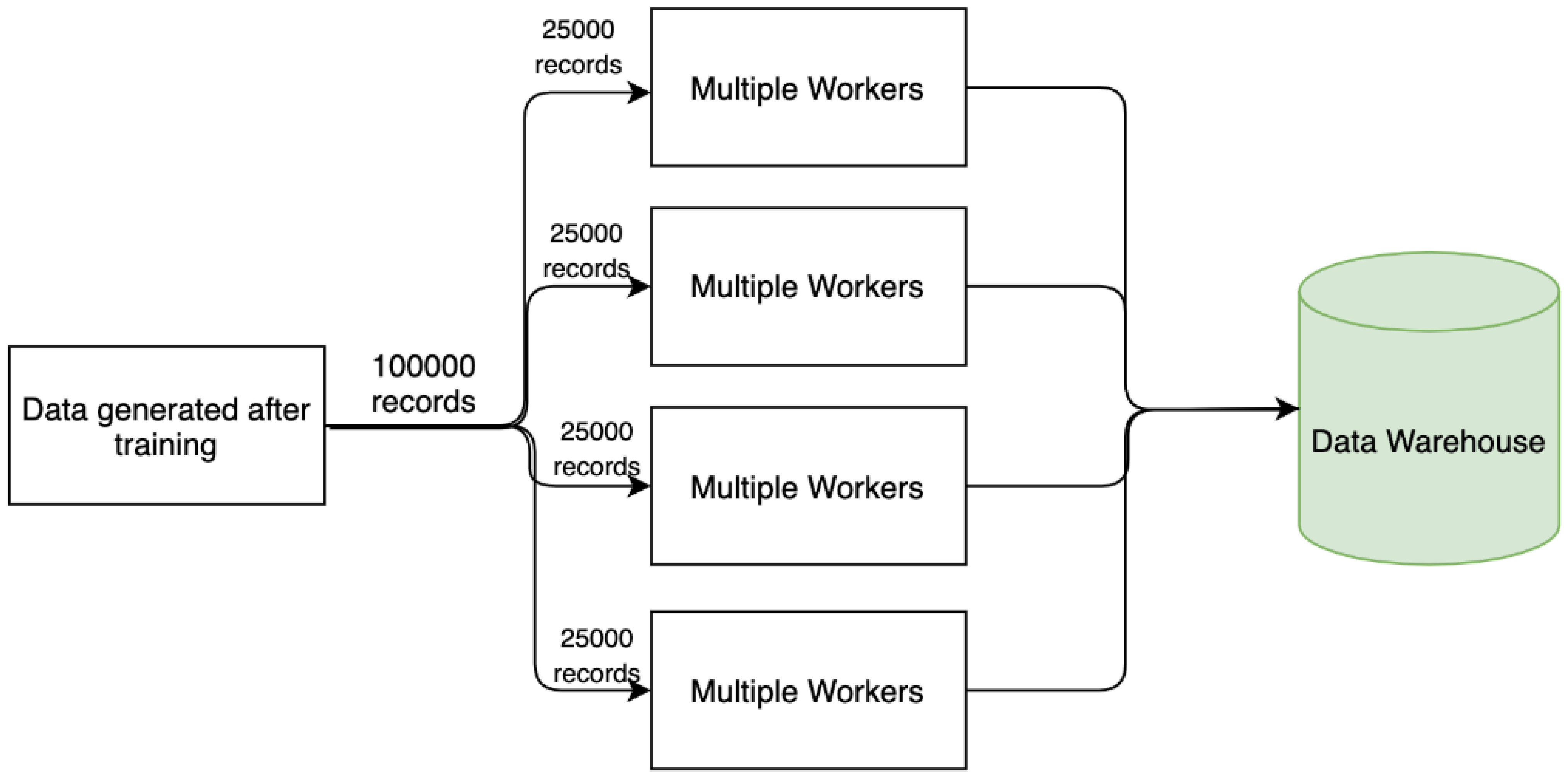
Parallel Processing

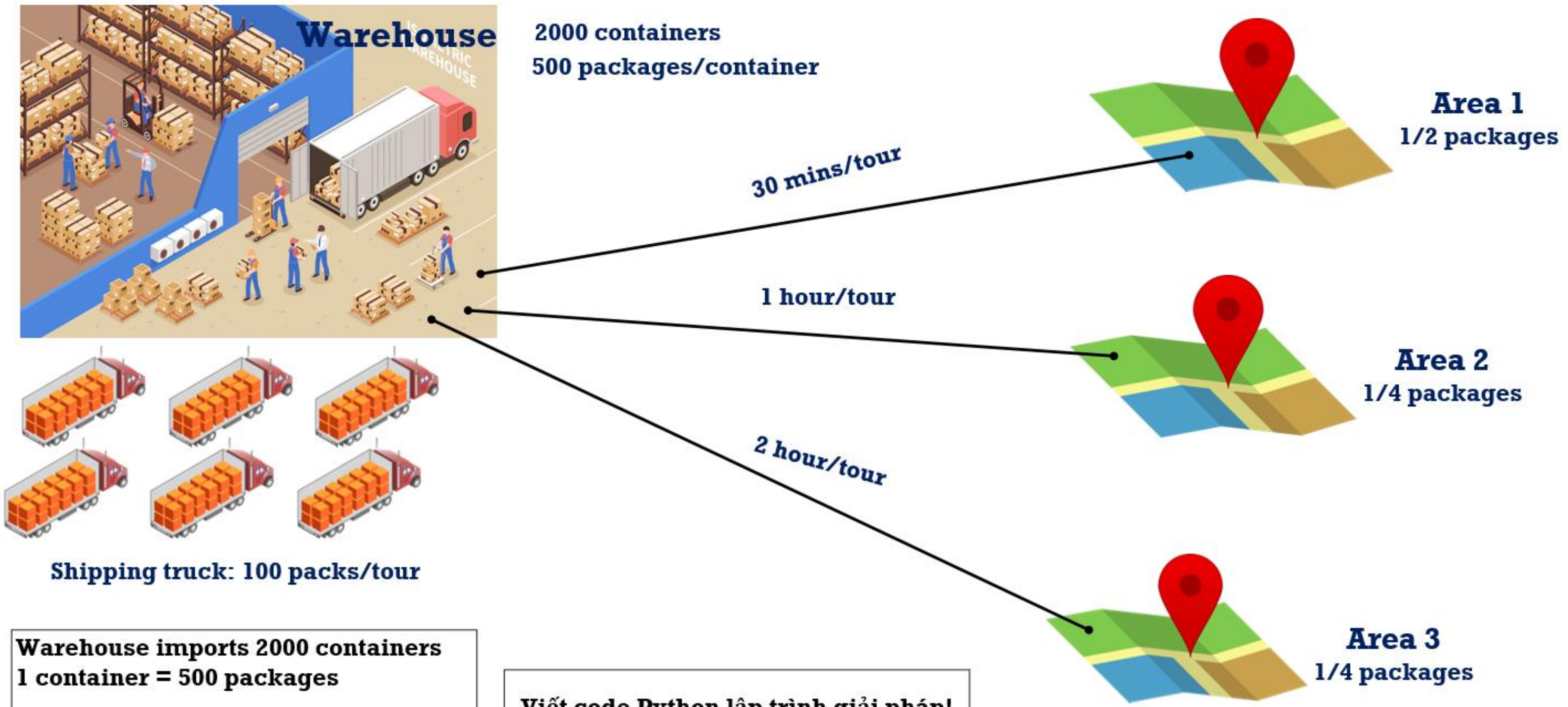
.



Parallel Processing

.





Warehouse imports 2000 containers
1 container = 500 packages

? Cần bao nhiêu trucks để vận chuyển hết hàng hóa trong vòng 1 tuần

Viết code Python lập trình giải pháp!

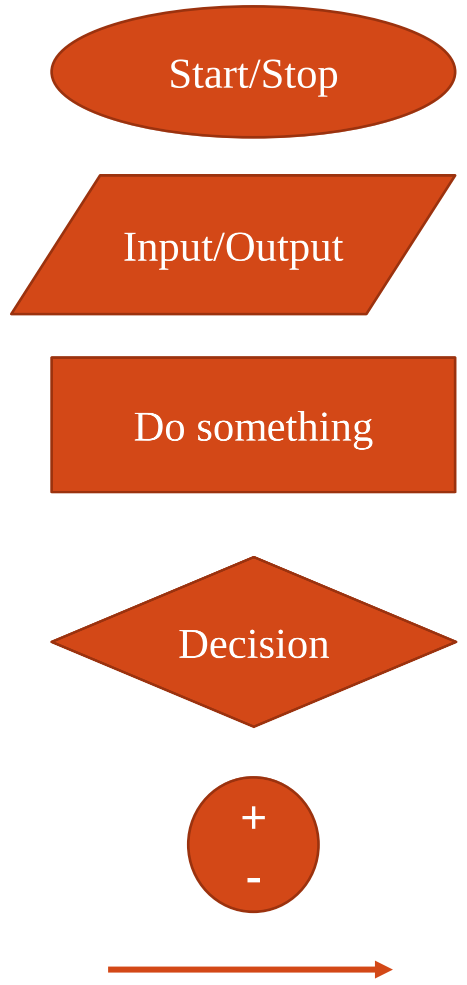
Practices (Parallel processing)

❑ Xem xét model:

Xây dựng mô hình toán mô tả và đề xuất giải pháp.

Vẽ flowchart lập trình chương trình cho Python.

Viết code python thực thi giải thuật/giải pháp đã đề xuất.


Start/Stop
Input/Output
Do something
Decision
+
-