Python Generator Genius

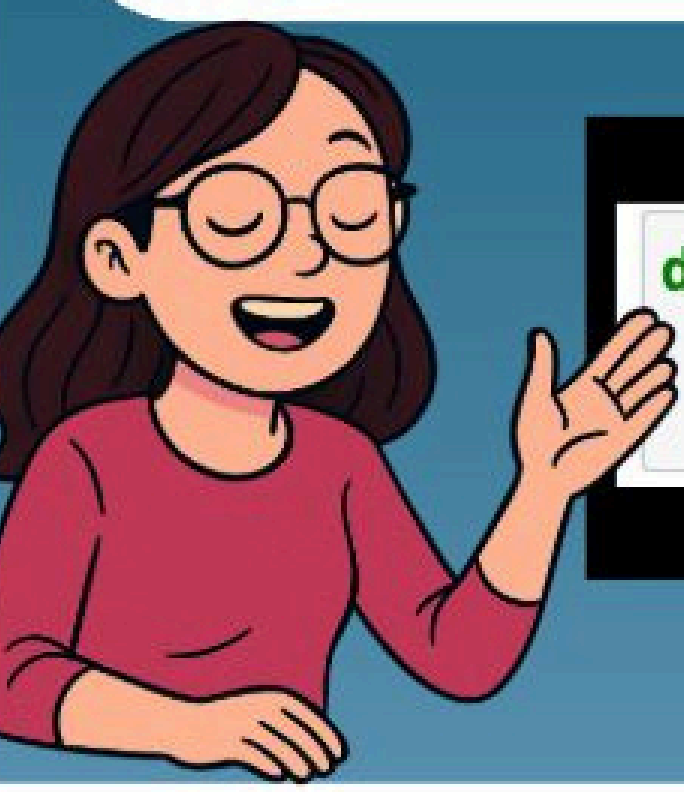
*Interviewer

*Bhawana

learnwithbhawana


Alright Bhawana!! Let's dive into Generators
Can you explain what is Generator in Python?
Sure !!
learnwithbhawana

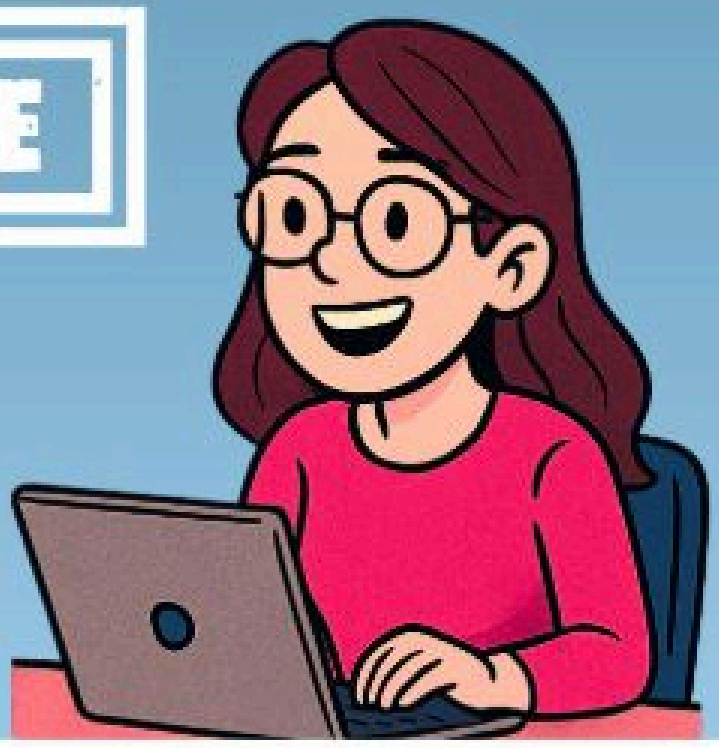
- A Generator is a special type of Iterable,like Lists or Tuples.
- Generators creates Values On-the-fly using yield.

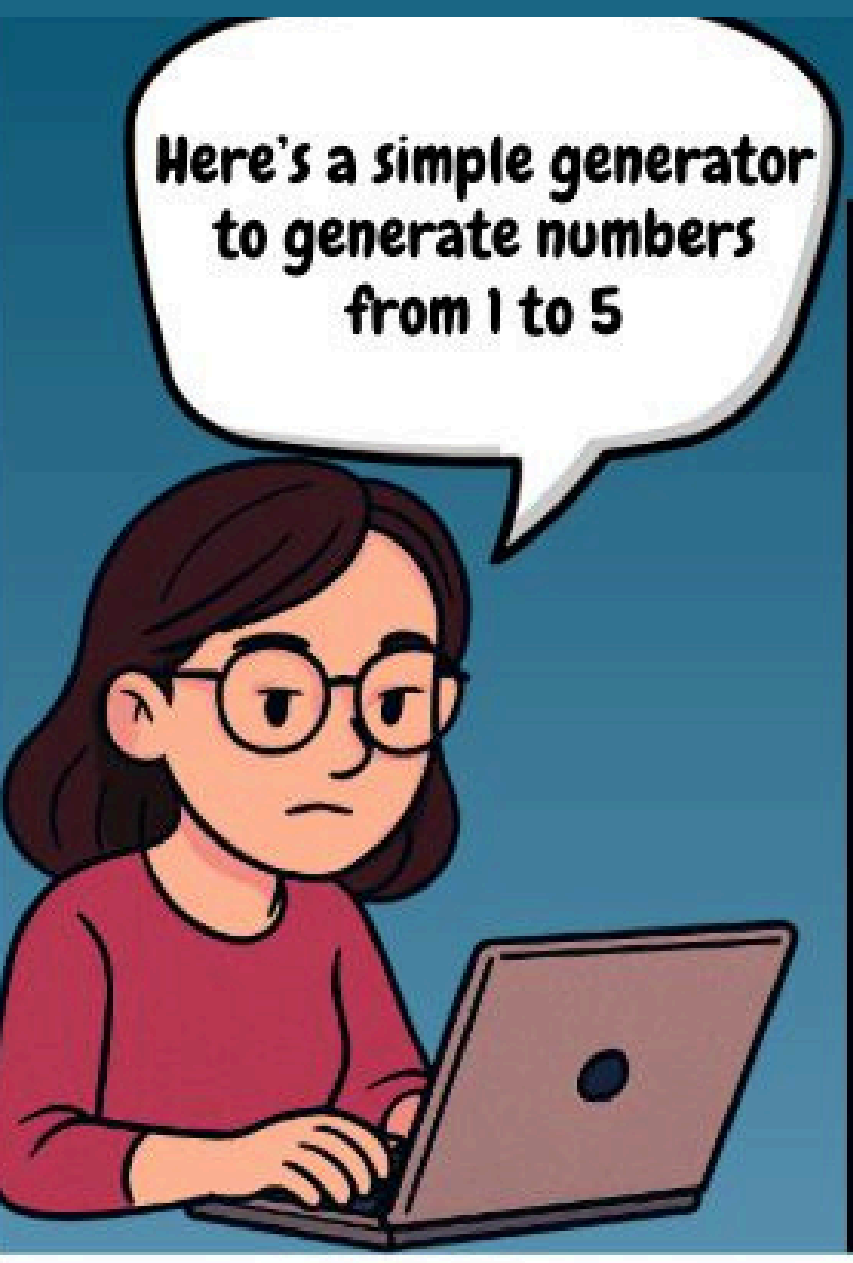


```python
def generator_function():
    # Logic
    yield value
```

Correct.
Can you give an example?

EXAMPLE





Code

```
def gen_numbers(n):
    for i in range(1,n+1):
        yield i

#Using the generator

for num in gen_numbers(5):
    print(num)
```

O/p:

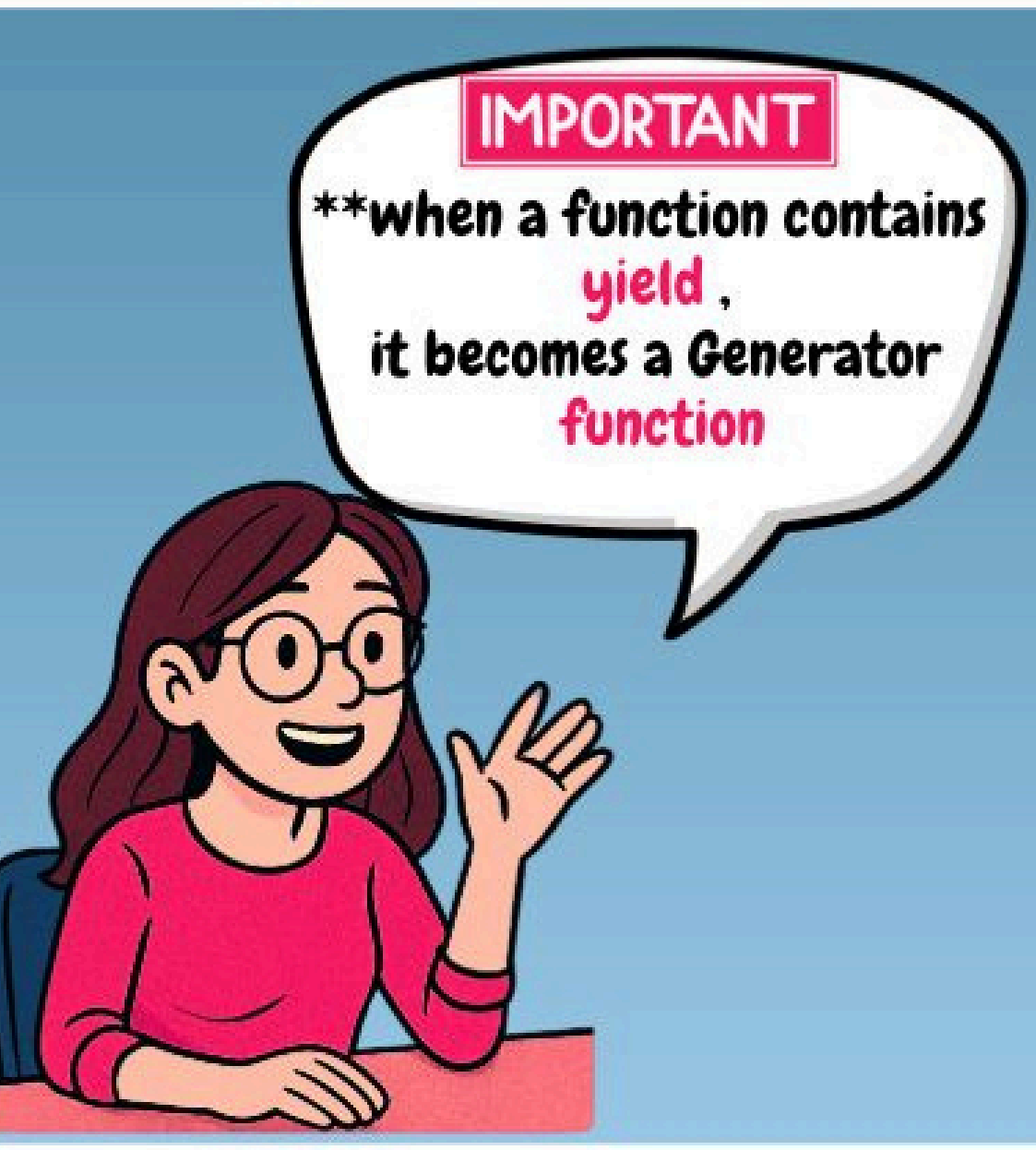
```
1
2
3
4
5
```

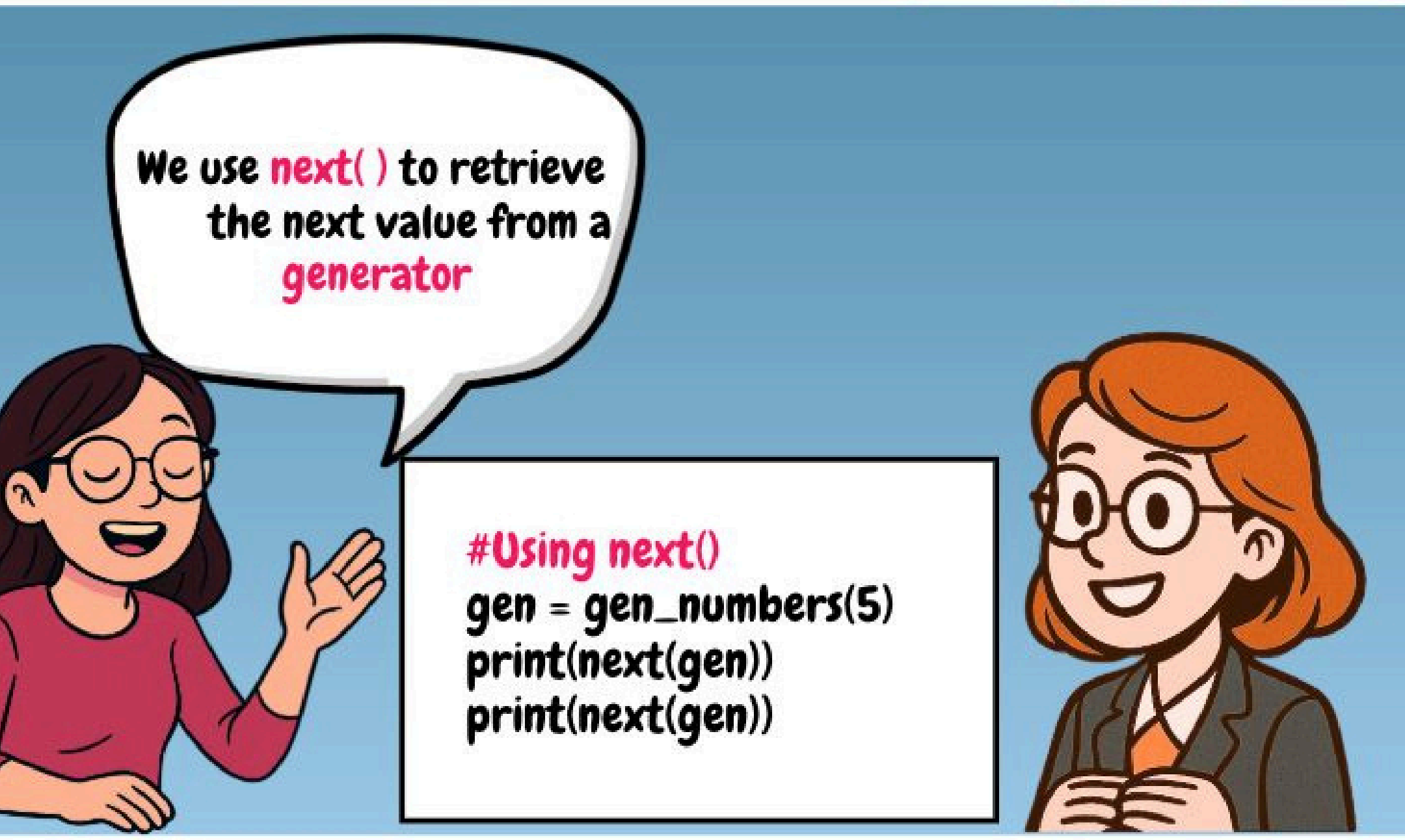
return

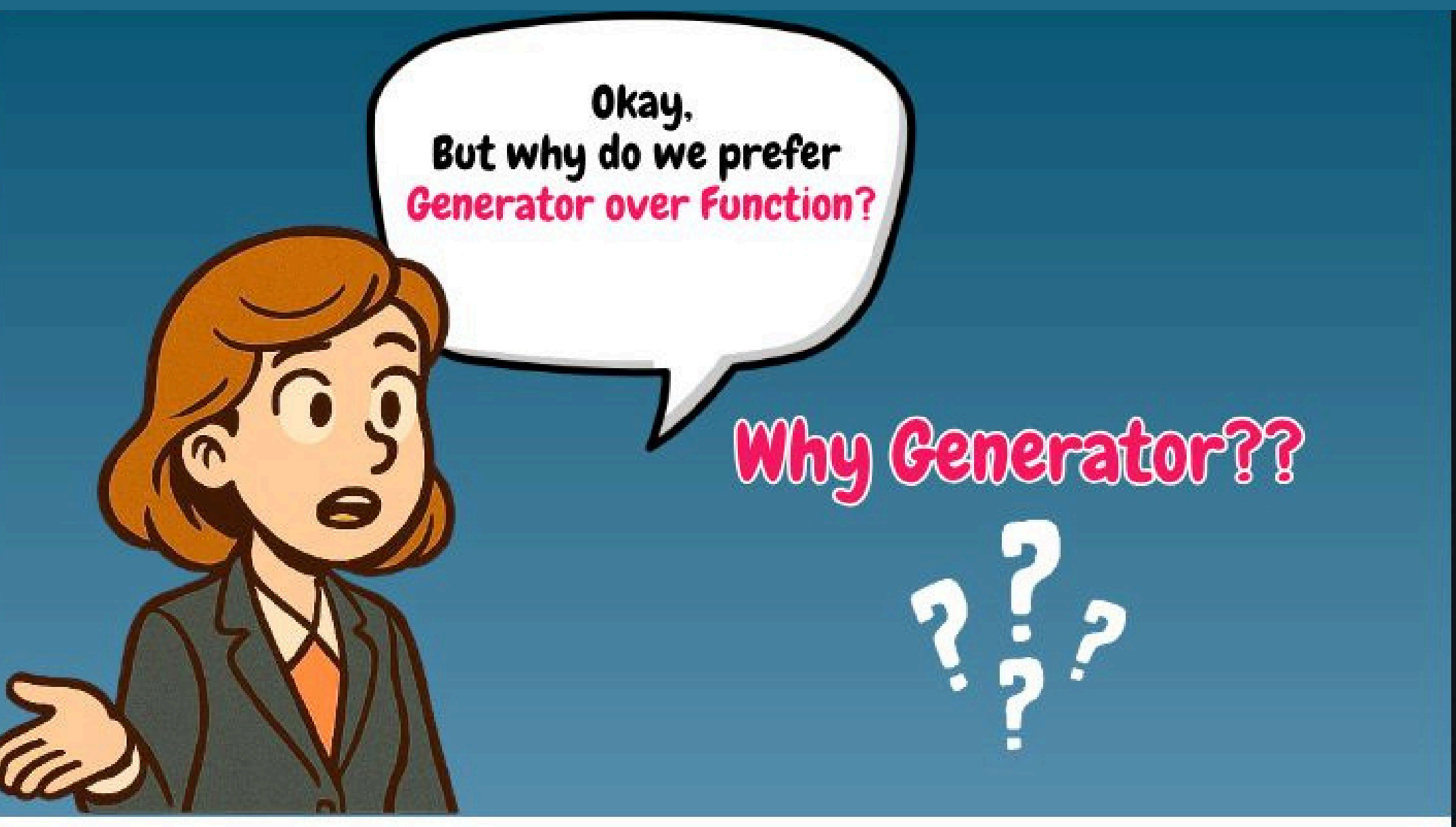
- terminates function execution
- returns a value(cannot be resumed)

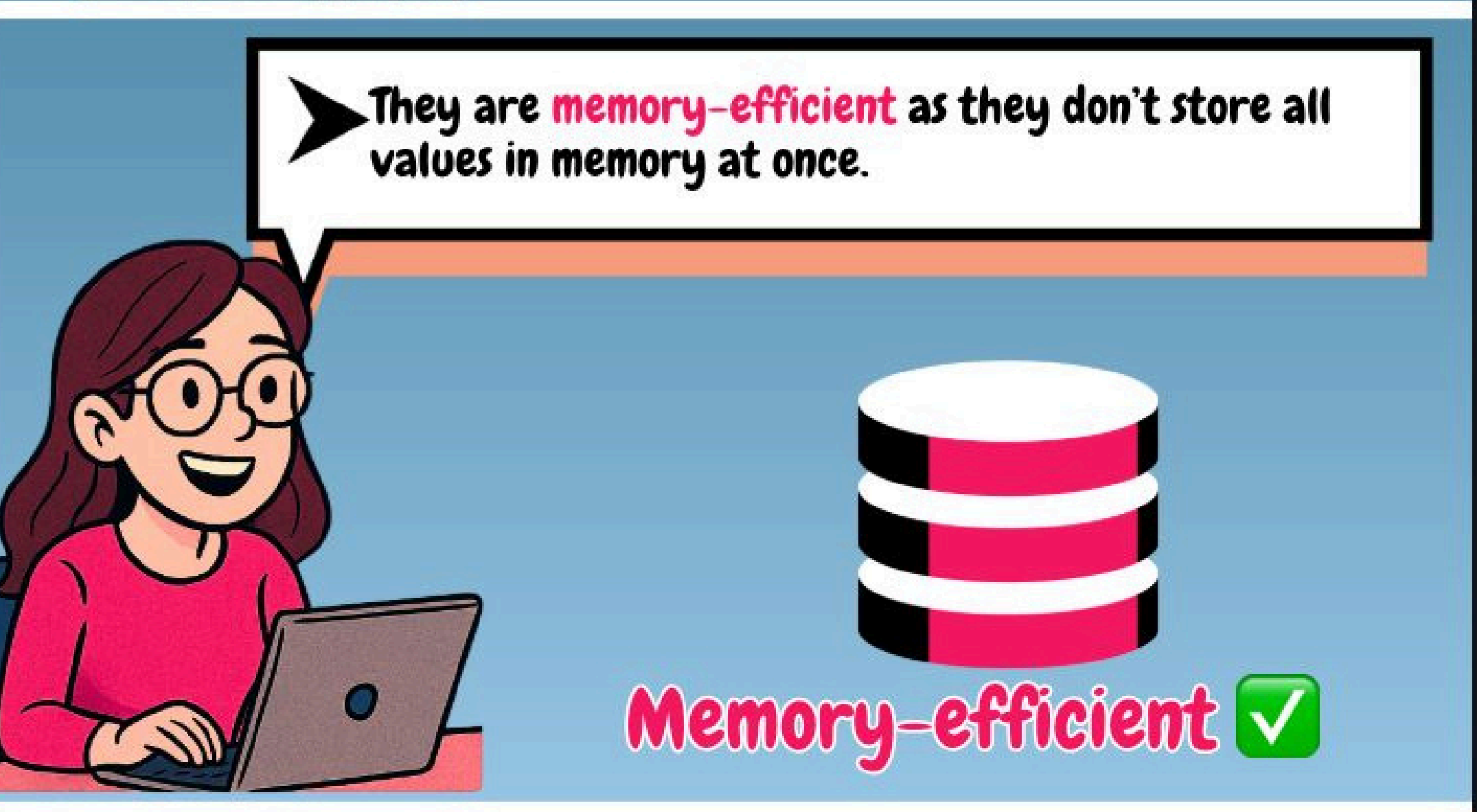
yield

- Pauses function execution
- returns a value,resumes where it left-off

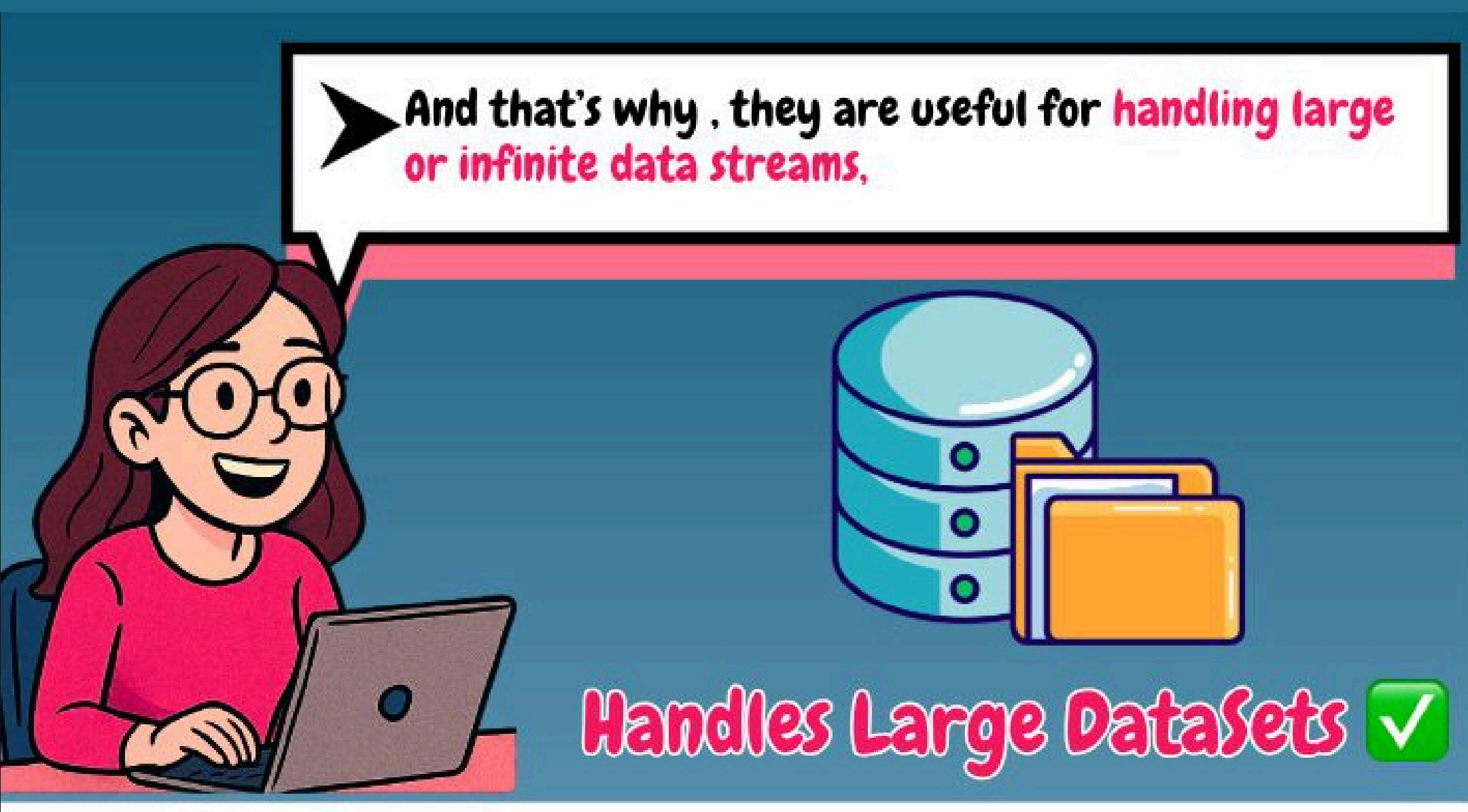


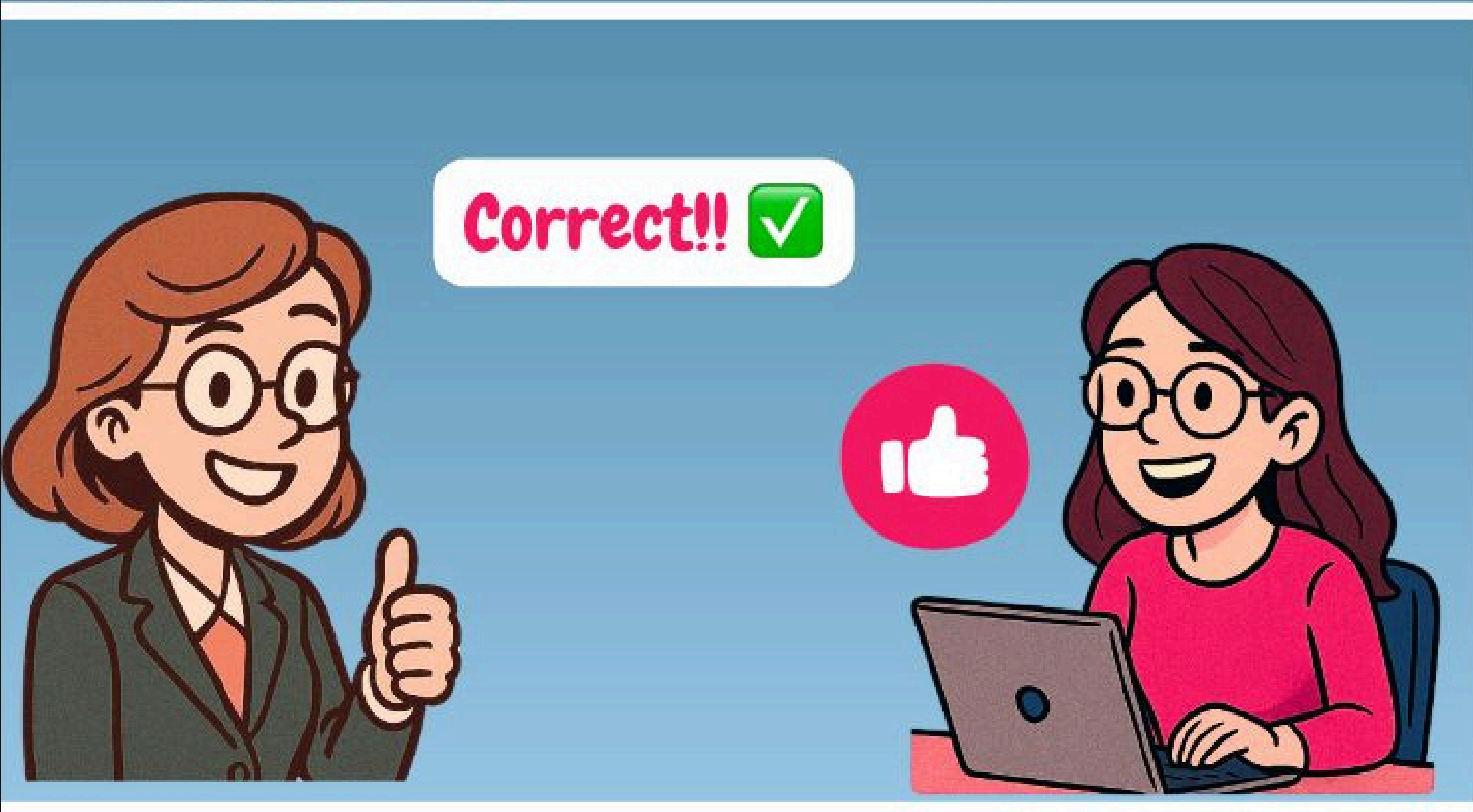






learnwithbhawana



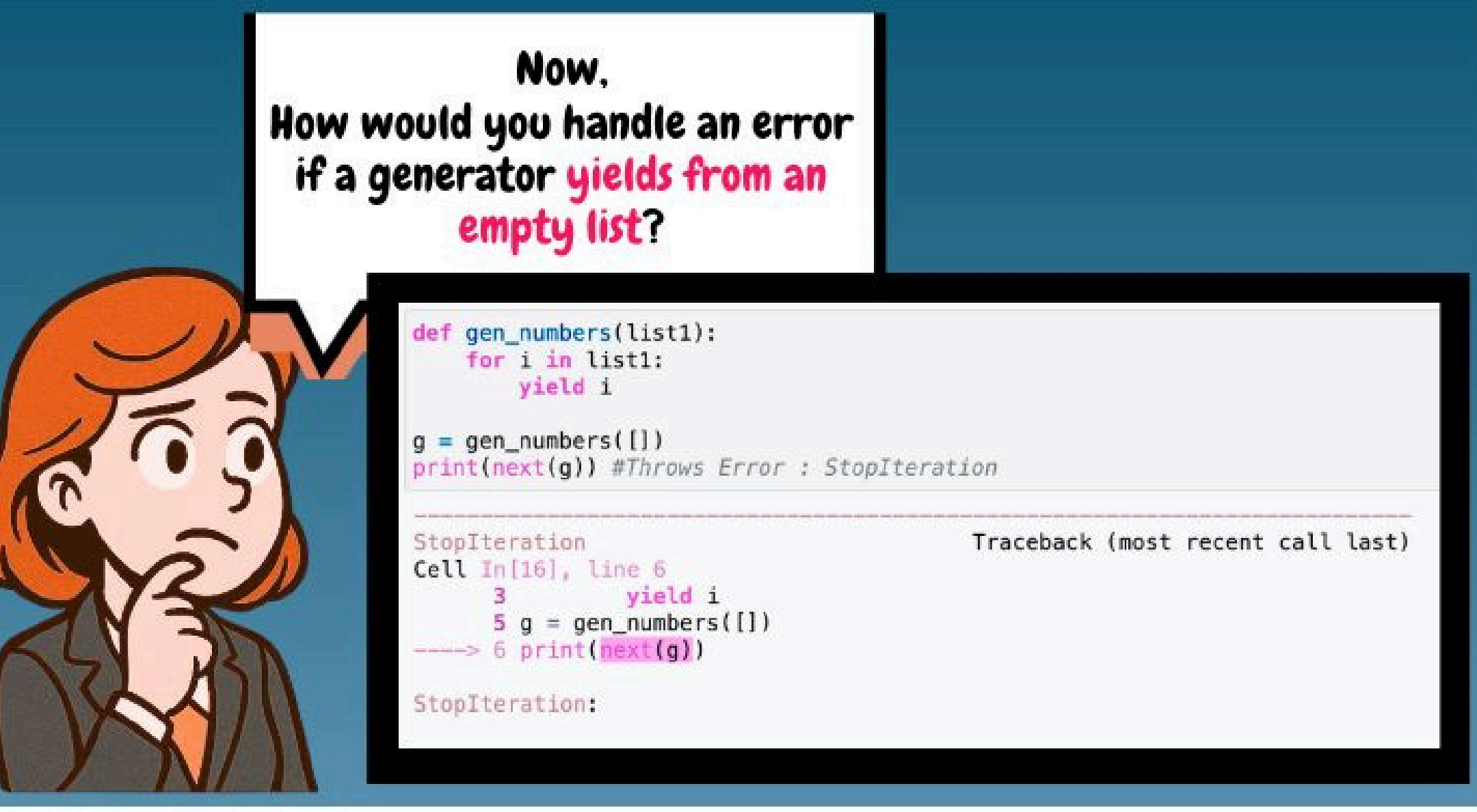


Stopiteration Exception

```
def gen_numbers():
    yield 1

g = gen_numbers()
print(next(g)) #Output : 1
print(next(g)) #StopIteration Error
```

```
1

---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
Cell In[14], line 6
      4 g = gen_numbers()
      5 print(next(g)) #Output : 1
----> 6 print(next(g))

StopIteration:
```

learnwithbhawana



learnwithbhawana

Option 1 : Check before creating the generator!

```python
def gen_numbers(list1):
    if not list1:
        print("List is Empty!!")
        return
    for i in list1:
        yield i

g = gen_numbers([])
print(next(g))
```

If condition to ensure the list is not empty

```
List is Empty!!

---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
Cell In[20], line 9
      6         yield i
      8 g = gen_numbers([])
----> 9 print(next(g))

StopIteration:
```

Option 2 : Catch StopIteration using try-except

```python
def gen_numbers(list1):
    for i in list1:
        yield i

try:
    g = gen_numbers([])
    print(next(g))
except StopIteration:
    print("No values to yield!!")
```

Except block to handle StopIteration

```
No values to yield!!
```

Great Job.Last question:
What do send(), throw(), and close() do in generators?

send( )

throw( )

close( )

send() sends a value to pause yield!

```python
def gen():
    while True:                 # Infinite loop
        val = yield             # Pause & wait
        print("Got:", val)      # Print value

g = gen()                       # Create generator
next(g)                         # Start generator
g.send(42)                      # Send value 42
```

```
Got: 42
```

throw() raises an exception inside generator!

```python
def gen():
    try:                                #try block
        yield 1                         #Yield value 1
    except ValueError:                  #Catch ValueError using except block
        print("ValueError caught!")     # Print message
        yield                           # Resume yielding

g = gen()                               # Create generator
next(g)                                 # Start generator
g.throw(ValueError)                     # Throw error in generator
```

```
ValueError caught!
```

close() stops the generator gracefully!

```python
def gen():
    try:                        # start try block
        while True:             # infinite loop
            yield               # yield control
    finally:                    # on exit block
        print("Closed!")        # cleanup message

g = gen()                       # create generator
next(g)                         # start generator
g.close()                       # close generator
```

```
Closed!
```

learnwithbhawana

learnwithbhawana