Varying Arguments

In Python we can pass different number of arguments of same type while invoking functions. Let's talk about Varying arguments.

- At times we might want to pass values of same type as arguments and we might not know how many of them.
- In that scenario we can leverage the concept of Varying arguments.
- The parameter which accepts Varying arguments should have `*` at the beginning - example: `*phone_numbers`.
- As part of the function body the type of the parameter will be `tuple`. In our case, as we will be passing phone numbers as strings, it will be of type `tuple` where each element will be of type string.

```python
def get_invalid_phone_count(*phone_numbers, employee_id):
    invalid_count = 0
    for phone_number in phone_numbers:
        if len(phone_number) < 10:
            invalid_count += 1
    return employee_id, invalid_count
```

```python
s = 'Employee {employee_id} have {invalid_count} invalid phones'
employee_id, invalid_count = get_invalid_phone_count('1234', '1234567890', 1)
# argument by position will fail
# Python interpreter cannot determine whether 1 is related to phone_numbers or employee_id
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-0fa187d3d10e> in <module>
      1 s = 'Employee {employee_id} have {invalid_count} invalid phones'
----> 2 employee_id, invalid_count = get_invalid_phone_count('1234', '1234567890', 1)
      3 # argument by position will fail
      4 # Python interpreter cannot determine whether 1 is related to phone_numbers or 
employee_id

TypeError: get_invalid_phone_count() missing 1 required keyword-only argument: 'employee_id'
```

```python
s = 'Employee {employee_id} have {invalid_count} invalid phones'
phone_numbers = ('1234', '1234567890',)
employee_id, invalid_count = get_invalid_phone_count(employee_id=1, 
phone_numbers=phone_numbers)
# argument by name will fail
# This will fail as we cannot pass varrying argument using keyword
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-05a120172b4e> in <module>
      1 s = 'Employee {employee_id} have {invalid_count} invalid phones'
      2 phone_numbers = ('1234', '1234567890',)
----> 3 employee_id, invalid_count = get_invalid_phone_count(employee_id=1, 
phone_numbers=phone_numbers)
      4 # argument by name will fail
      5 # This will fail as we cannot pass varrying argument using keyword

TypeError: get_invalid_phone_count() got an unexpected keyword argument 'phone_numbers'
```

```python
s = 'Employee {employee_id} have {invalid_count} invalid phones'
employee_id, invalid_count = get_invalid_phone_count('1234', '1234567890', employee_id=1)
# argument by position will fail
# Python interpreter cannot determine whether 1 is related to phone_numbers or employee_id
```

```python
print(s.format(employee_id=employee_id, invalid_count=invalid_count))
```

```
Employee 1 have 1 invalid phones
```

```python
def get_invalid_phone_count(employee_id, *phone_numbers):
    print(f'Length of phone_numbers is: {len(phone_numbers)}')
    print(f'Type of phone_numbers is: {type(phone_numbers)}')
    print(f'Type of each phone number is: {type(phone_numbers[0])}')
    print(phone_numbers)
    invalid_count = 0
    for phone_number in phone_numbers:
        if len(phone_number) < 10:
            invalid_count += 1
    return employee_id, invalid_count
```

```python
s = 'Employee {employee_id} have {invalid_count} invalid phones'
employee_id, invalid_count = get_invalid_phone_count(1, '1234', '1234567890') # argument by position works here
```

```
Length of phone_numbers is: 2
Type of phone_numbers is: <class 'tuple'>
Type of each phone number is: <class 'str'>
('1234', '1234567890')
```

```python
print(s.format(employee_id=employee_id, invalid_count=invalid_count))
```

```
Employee 1 have 1 invalid phones
```

```python
s = 'Employee {employee_id} have {invalid_count} invalid phones'
phone_numbers = ['1234', '1234567890']
```

```python
employee_id, invalid_count = get_invalid_phone_count(1, *phone_numbers) # argument by position works here
```

```
Length of phone_numbers is: 2
Type of phone_numbers is: <class 'tuple'>
Type of each phone number is: <class 'str'>
('1234', '1234567890')
```

```python
print(s.format(employee_id=employee_id, invalid_count=invalid_count))
```

```
Employee 1 have 1 invalid phones
```