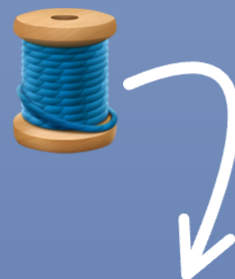




***args & *kwargs clearly
explained!! 🚀**



`*args` allows you to pass a variable number of non-keyword arguments to a function.


It collects all non-keyword arguments passed to the function and stores them as a tuple.

Consider the following example:

Swipe 



@akshay_pachaar



```
def print_args(*args):
```

```
    for arg in args:
```

```
        print(arg)
```

A tuple

```
print_args(1, 2, 3)
```

```
# 1
```

```
# 2
```

```
# 3
```

*Diff number of
arguments*

```
print_args(1, 2, 3, 4)
```

```
# 1
```

```
# 2
```

```
# 3
```

```
# 4
```

Similarly, `**kwargs` allows you to pass a variable number of keyword arguments to a function.

It collects all keyword arguments passed to the function and stores them as a dictionary.

Consider the following example:

Swipe 




@akshay_pachaar



```
def print_kwargs(**kwargs):  
    for key, value in kwargs.items():  
        print(f"{key}: {value}")
```

A dictionary




```
print_kwargs(a=1, b=2, c=3)
```

```
# a: 1
```

```
# b: 2
```


```
# c: 3
```

 @akshay_pachaar

You can also use `*args` and `**kwargs` together in a function definition.

Check this:


Swipe 



```
def print_args_kwargs(*args, **kwargs):  
    for arg in args:  
        print(f"Non-keyword argument: {arg}")  
    for key, value in kwargs.items():  
        print(f"Keyword argument: {key}={value}")
```

```
print_args_kwargs(1, 2, 3, a=4, b=5, c=6)
```

```
# Non-keyword argument: 1  
# Non-keyword argument: 2  
# Non-keyword argument: 3  
# Keyword argument: a=4  
# Keyword argument: b=5  
# Keyword argument: c=6
```

 @akshay_pachaar

It's important to note that the order of parameters in the function definition matters.

The syntax for using `*args` and `**kwargs` is:

Swipe 

Order matters!! ✓



correct.py

```
# correct_function_definition.py
def my_function(normal_args, *args, **kwargs):
    pass
```










incorrect.py

```
# wrong_function_definition.py
def my_function(normal_args, **kwargs, *args):
    pass
```

That's a wrap!

If you interested in:

- Python 
- Data Science 
- Machine Learning 
- MLOps 
- NLP 
- Computer Vision 
- LLMs 

Follow me on LinkedIn 

Everyday, I share tutorials on above topics!

Cheers!! 



@akshay_pachaar