*ARGS & **KWARGS

```
def myfunc(a=0,b=0,c=0,d=0,e=0):
    return sum((a,b,c,d,e))*0.05

myfunc(40,60,20)
```

o/p: 6.0

Hence, this isn't a effecient soln.

We use *args.

When a function parameter starts with an asterisk, it allows for an arbitrary no. of arguments, & the function takes them in as a tuple of values.

```
In: def func(*args):
        return sum(args)*.05

myfunc(40,60,20)
```

O/P: 6

Note: Passing the keyword "args" into sum() function did the same thing as a tuple of arguments. It is worth nothing that the word "args" is only arbitrary — any word would do so as long as it's preceded by an asterisk.

```
In: def myfunc(*spam):
        return sum(spam)*.05

myfunc(40,60,20)
```

O/P: 6.0

**KWARGS

It handles keyworded arguments. Instead of creating a tuple. It creates a dictionary of key/value pairs.

```
In: def myfunc(**kwargs):
        if 'fruit' in kwargs:
            print(f"My fav. fruit is {kwargs['fruit']}")
        else:
            print("I don't like fruit")

In: myfunc(fruit="pineapple")

O/p: My fav. fruit is pineapple

In: myfunc()

O/p: I don't like fruit
```

*args & **KWARGS

You can pass *args & **kwargs into the same function, but *args must appear before **kwargs.

```
In: def myfunc(*args, **kwargs):
        if 'fruit' & 'juice' in kwargs:
            print(f"I like {' and '.join(args)} &
            my favourite fruit is {kwargs['fruit']}")
```

print(f"May I have some {kwargs['juice']} juice?")

else:
    pass

myfunc('eggs', 'spam', fruit='cherries', juice='orange')

o/p: I like eggs & spam & my favourite fruit is cherries
May I have some orange juice?

Note: Placing keyworded arguments ahead of positional arguments raises an exception.

In: myfunc(fruit='cherries', juice='orange', 'eggs', 'spam')

o/p: Syntax Error.