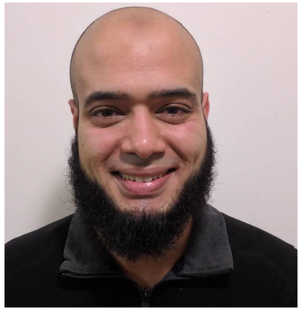
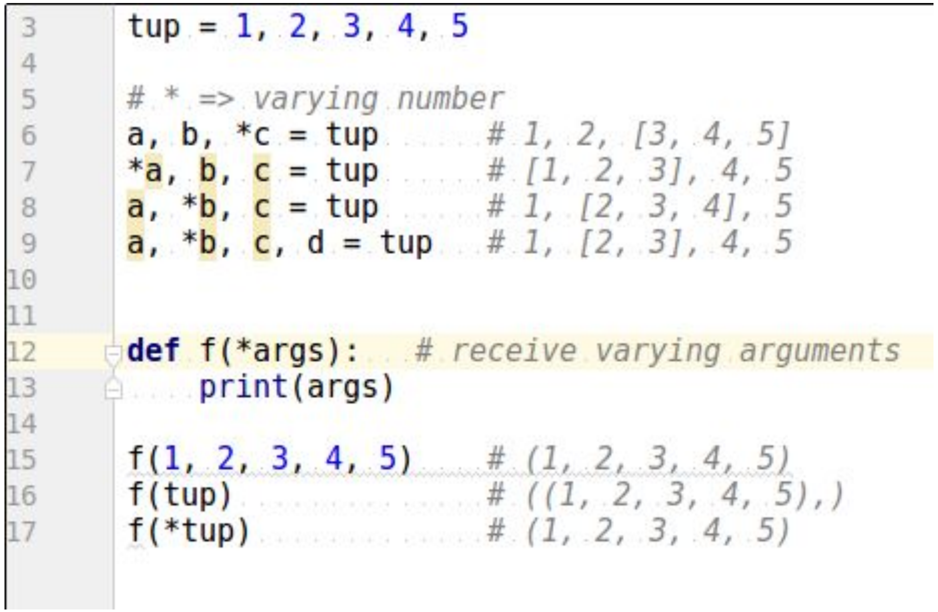
Python Programming
*args and **kwargs

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

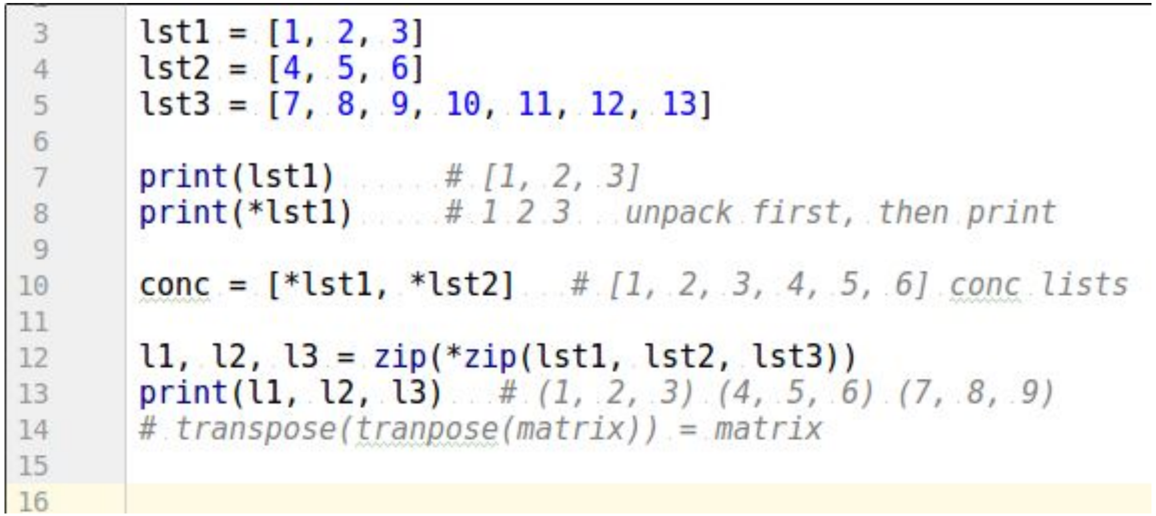


*args (positional expansion)



```python
tup = 1, 2, 3, 4, 5

# * => varying number
a, b, *c = tup        # 1, 2, [3, 4, 5]
*a, b, c = tup        # [1, 2, 3], 4, 5
a, *b, c = tup        # 1, [2, 3, 4], 5
a, *b, c, d = tup     # 1, [2, 3], 4, 5


def f(*args):    # receive varying arguments
    print(args)

f(1, 2, 3, 4, 5)      # (1, 2, 3, 4, 5)
f(tup)                # ((1, 2, 3, 4, 5),)
f(*tup)               # (1, 2, 3, 4, 5)
```

*args



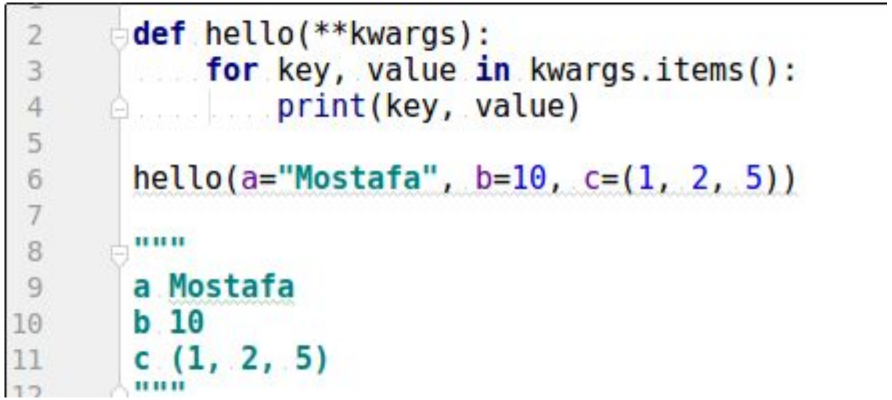
```
lst1 = [1, 2, 3]
lst2 = [4, 5, 6]
lst3 = [7, 8, 9, 10, 11, 12, 13]

print(lst1)       # [1, 2, 3]
print(*lst1)      # 1 2 3   unpack first, then print

conc = [*lst1, *lst2]   # [1, 2, 3, 4, 5, 6] conc lists

l1, l2, l3 = zip(*zip(lst1, lst2, lst3))
print(l1, l2, l3)   # (1, 2, 3) (4, 5, 6) (7, 8, 9)
# transpose(tranpose(matrix)) = matrix
```

**kwargs (Keyword Arguments / keyword expansion)

- **kwargs is similar to *args
    - * args accepts **positional** arguments (**tuple** of items)
    - **kwargs accepts **keyword** (**dict** of items)
- Reading: What to call them? Eg. Splat in Ruby community

**kwargs

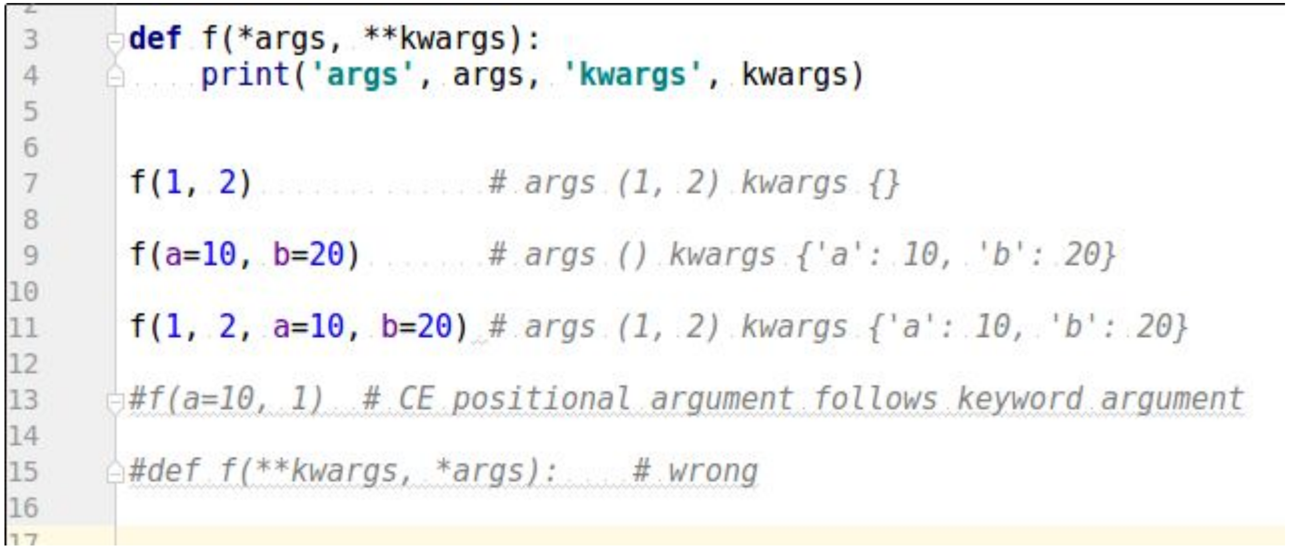
- We can pass a varying number of keywords
- All of the passed items will be as a dictionary
    - name = ‘mostafa’
    - Name will be a key (string)
    - ‘mostafa’ will be a value



```python
def hello(**kwargs):
    for key, value in kwargs.items():
        print(key, value)

hello(a="Mostafa", b=10, c=(1, 2, 5))

"""
a Mostafa
b 10
c (1, 2, 5)
"""
```

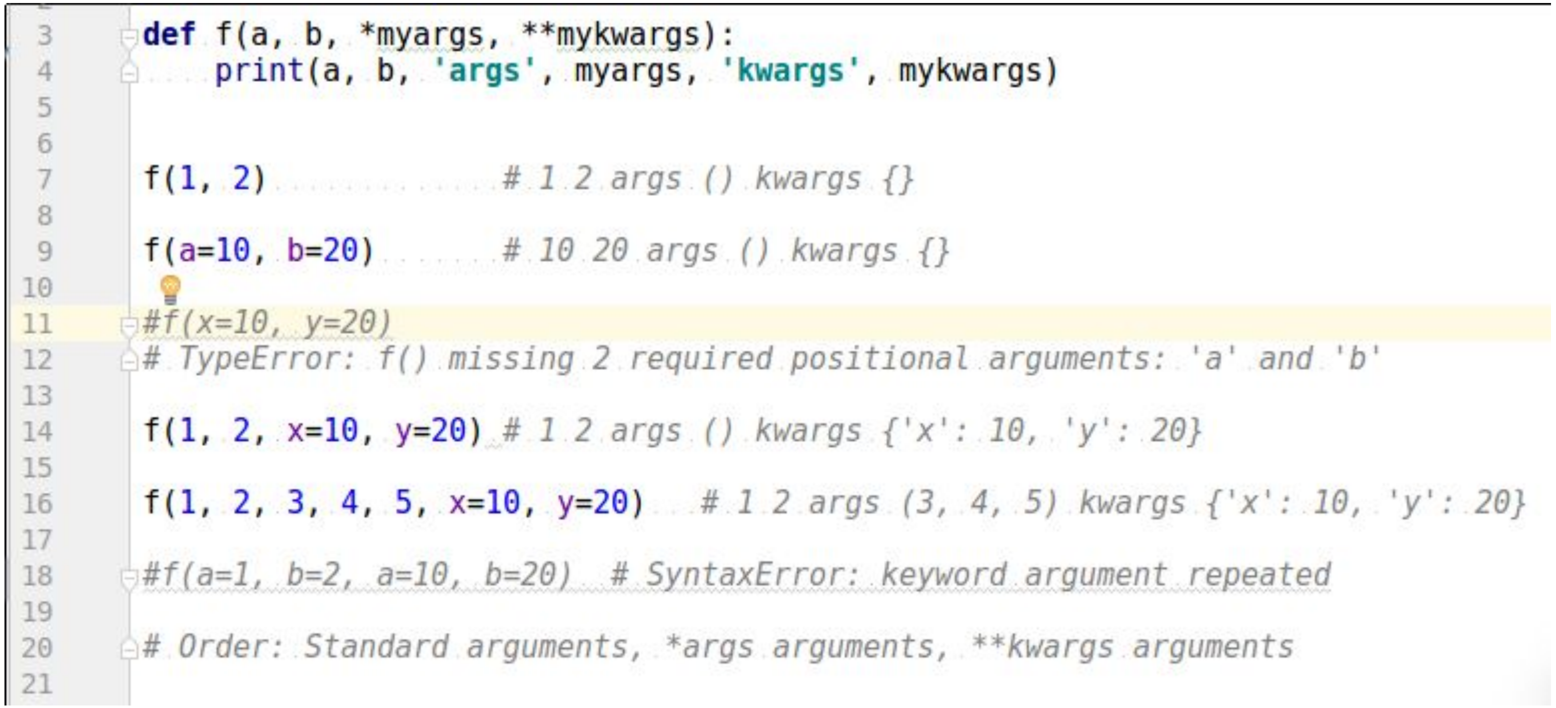
*args, **kwargs

- We can have both of them together, but respect the order



```python
def f(*args, **kwargs):
    print('args', args, 'kwargs', kwargs)


f(1, 2)                # args (1, 2) kwargs {}

f(a=10, b=20)          # args () kwargs {'a': 10, 'b': 20}

f(1, 2, a=10, b=20) # args (1, 2) kwargs {'a': 10, 'b': 20}

#f(a=10, 1)  # CE positional argument follows keyword argument

#def f(**kwargs, *args):    # wrong
```

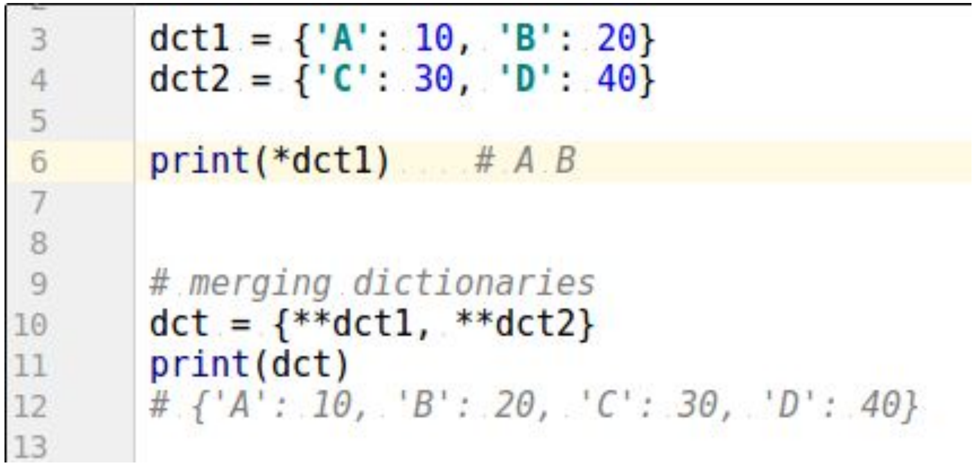
Standard, positional and keyword arguments



```python
def f(a, b, *myargs, **mykwargs):
    print(a, b, 'args', myargs, 'kwargs', mykwargs)


f(1, 2)                 # 1 2 args () kwargs {}

f(a=10, b=20)           # 10 20 args () kwargs {}

#f(x=10, y=20)
# TypeError: f() missing 2 required positional arguments: 'a' and 'b'

f(1, 2, x=10, y=20) # 1 2 args () kwargs {'x': 10, 'y': 20}

f(1, 2, 3, 4, 5, x=10, y=20)   # 1 2 args (3, 4, 5) kwargs {'x': 10, 'y': 20}

#f(a=1, b=2, a=10, b=20)  # SyntaxError: keyword argument repeated

# Order: Standard arguments, *args arguments, **kwargs arguments

```

Merging Dictionaries

- **dict will expand to its tuple of (key, value), hence we can build new dict from it



```python
dct1 = {'A': 10, 'B': 20}
dct2 = {'C': 30, 'D': 40}

print(*dct1)    # A B


# merging dictionaries
dct = {**dct1, **dct2}
print(dct)
# {'A': 10, 'B': 20, 'C': 30, 'D': 40}

```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."