Advanced Parameter Passing: *args and **kwargs in Python!

Hey Python superstars! Today, we're leveling up with **advanced parameter passing**—handling tons of arguments like pros. We'll explore:

- `*args` : Grab all the extras in a list.
- `**kwargs` : Scoop up named extras in a dictionary.
- Mix them with regular parameters!

Run the cells, tweak the code, and tackle the challenges—let's juggle arguments!

What's *args?

`*args` lets a function take any number of positional arguments, packing them into a tuple. It's like saying, "Give me all you've got!"

```
# Example 1: Basic *args
def add_all(*args):
    print(args)
    print(type(args))
    total = 0
    for num in args:
        total += num
    return total
```

```
print(add_all(1, 2, 3))
```

```
(1, 2, 3)
<class 'tuple'>
6
```

```


print(add_all(1))
```

```
(1,)
<class 'tuple'>
1
```

```
# Example 2: With a Fixed Parameter
def greet_group(leader, *group):
    print(f"Leader: {leader}")
    print(f"Group: {group}")
```

```
greet_group("Anna", "Bob", "Cleo", "Dan")
```

```
Leader: Anna
Group: ('Bob', 'Cleo', 'Dan')
```

```
# Challenge: Count your args!
def list_items(*things):
    print(f"You gave me {len(things)} things:")
    for item in things:
        print(f"- {item}")
```

```
list_items("apple", "banana", "grape")
# Try 2 items or 5—what's the count?
```

```
You gave me 3 things:
- apple
- banana
- grape
```

What's **kwargs?

`**kwargs` grabs any number of keyword arguments, stuffing them into a dictionary. It's like collecting labeled notes!

```
# Example 3: Basic **kwargs
def show_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")
```

```
show_info(name="Zoe", age=15, hobby="dance")
```

```
name: Zoe
age: 15
hobby: dance
```

```
# Example 4: With a Fixed Parameter
def describe_pet(pet_type, **details):
    print(f"Pet type: {pet_type}")
    for key, value in details.items():
        print(f"{key}: {value}")
```

```
describe_pet("dog", name="Max", color="brown")
```

```
{'name': 'Max', 'color': 'brown'}
{'name': 'Max', 'color': 'brown'}
```

```

# Challenge: Describe yourself!
def about_me(**info):
    print("All about me:")
    for k, v in info.items():
        print(f"- {k}: {v}")
```

```
about_me(name="Sam", favorite="pizza")
```

```
All about me:
- name: Sam
- favorite: pizza
```

Mixing It All: Regular, *args, *kwargs

You can combine regular parameters, `*args`, and `**kwargs`—but order matters:

- Regular first, then `*args`, then `**kwargs`. Let's see the magic!

```
# Example 5: All Together
def party_plan(theme, *guests, **details):
    print(f"Theme: {theme}")
    print(f"Guests: {guests}")
    print("Details:")
    for key, value in details.items():
        print(f"- {key}: {value}")
```

```
party_plan("beach", "Tina", "Leo", "Mia", time="6pm", food="BBQ")
```

```
Theme: beach
Guests: ('Tina', 'Leo', 'Mia')
Details:
- time: 6pm
- food: BBQ
```

```
# Challenge: Plan your event!
def event(main_item, *people, **extras):
    print(f"Main item: {main_item}")
    print(f"People: {people}")
    print("Extras:")
```

```
    for k, v in extras.items():
        print(f"- {k}: {v}")
```

```
event("cake", "Ali", "Ben", place="park", time="noon")
```

```
Main item: cake
People: ('Ali', 'Ben')
Extras:
- place: park
- time: noon
```

Unpacking with * and **: Reverse It!

You can *unpack* lists or dicts into arguments using `*` (for positional) and `**` (for keyword). It's like spilling the bag!

```
# Example 6: Unpacking *args
def multiply(a, b, c):
    return a * b * c

nums = [2, 3, 4]

```

```
print(multiply(*nums))
```

```
24
```

```
# Example 7: Unpacking **kwargs
def introduce(name, job):
    print(f"{name} is a {job}.")
```

```
info = {"name": "Elle", "job": "artist"}
introduce(**info)
```

```
Elle is a artist.
```

```
# Challenge: Unpack your own!
def add_three(x, y, z):
    return x + y + z
```

```
values = [1, 2, 3]
print(add_three(*values))
```

```
6
```

Interactive Challenges: Args and Kwargs Time!

Predict, run, and tweak these tricky ones!

```
# Challenge 1: Count Everything
def tally(name, *scores, **notes):
    print(f"Name: {name}")
    print(f"Total scores: {len(scores)}")
    print(f"Notes count: {len(notes)}")

```

```
tally("Jay", 90, 85, 88, comment="Great!", effort="A+")
```

```
Name: Jay
Total scores: 3
Notes count: 2
```

```
def xyz(a= 5):
  return print(a)
a = 10
xyz(a)
```

```
10
```

```
# Challenge 2: Flexible Order
def mix_it(first, *rest, last="end", **options):
    print(f"First: {first}")
    print(f"Rest: {rest}")
    print(f"Last: {last}")
    print(f"Options: {options}")
```

```
mix_it("start", 1, 2, 3, last="finish", extra="yes")
```

```
# Challenge 3: Unpack and Run
def trip(days, *stops, **plans):
    print(f"Days: {days}")
    print(f"Stops: {stops}")
    print(f"Plans: {plans}")
```

```
trip_data = [3]
stops_list = ["park", "zoo"]
plans_dict = {"food": "picnic", "time": "morning"}

```

```
1 trip(*trip_data, *stops_list, **plans_dict)
```

Wrap-Up: Args and Kwargs Wizards!

You've conquered:

- `*args` : All the extras in a tuple.
- `**kwargs` : Named extras in a dict.
- **Mixing**: Combine and unpack like champs.

Keep juggling! Add more args, mix kwargs, or unpack a list. What's your favorite trick? Share it in class!