2. توابع با تعداد متغیر ورودی متغیر (Arbitrary Arguments)

در پایتون، زمانی که می‌خواهید یک تابع تعریف کنید که تعداد آرگومان‌های ورودی آن ثابت نیست و می‌تواند تعداد متغیری از آرگومان‌ها را دریافت کند، از ویژگی‌هایی مانند `args*` و `kwargs**` استفاده می‌کنیم.

1. استفاده از `args*\`

پارامتر `args*` به شما این امکان را می‌دهد که تعداد نامحدودی از آرگومان‌ها را به یک تابع ارسال کنید. وقتی از `args*` در تعریف تابع استفاده می‌کنید، پایتون تمامی آرگومان‌های ارسالی را در قالب یک تاپل (tuple) قرار می‌دهد.

- **چطور کار می‌کند؟**
  وقتی تابعی با `args*` تعریف می‌شود، تمام آرگومان‌هایی که به آن تابع ارسال می‌شوند، به صورت یک تاپل در `args` قرار می‌گیرند.
- **نحوه استفاده از `args*\`:**
  برای استفاده از `args*`، کافی است در تعریف تابع، قبل از نام پارامتر از `*` استفاده کنید. سپس می‌توانید این پارامتر را همانند یک تاپل (tuple) در داخل تابع استفاده کنید.

مثال:

```
def add_numbers(*args):
    total = 0
    for num in args:
        total += num
    return total

# فراخوانی تابع با تعداد متغیر آرگومان‌ها
result = add_numbers(10, 20, 30)
print(result)
```

در اینجا:

- تابع `add_numbers` به تعداد نامحدودی از اعداد به عنوان آرگومان ورودی می‌پذیرد.
- در داخل تابع، با استفاده از یک حلقه، تمامی مقادیر موجود در `args` جمع می‌شوند.
- مقدار `total` که حاصل جمع است، به عنوان نتیجه برگشت داده می‌شود.

خروجی برنامه:

```
60
```

در این مثال، تعداد آرگومان‌ها متغیر است و می‌توانیم هر تعداد عدد را به تابع ارسال کنیم.

2. ارسال لیستی از مقادیر به تابع با استفاده از `args*\`

اگر یک لیست یا تاپل داشته باشید و بخواهید آن را به عنوان آرگومان به تابع ارسال کنید، می‌توانید از `args*` برای "گسترش" لیست یا تاپل و ارسال مقادیر آن به تابع استفاده کنید.

مثال:

```
def print_numbers(*args):
    for number in args:
        print(number)

numbers_list = [1, 2, 3, 4, 5]
print_numbers(*numbers_list)
```

در اینجا:

- از `numbers_list*` برای ارسال هر عدد موجود در لیست به عنوان آرگومان جداگانه به تابع `print_numbers` استفاده شده است.
- به این صورت که هر عدد داخل لیست به عنوان یک پارامتر مستقل به تابع ارسال می‌شود.

خروجی برنامه:

```
1
2
3
4
5
```

این نشان می‌دهد که چطور می‌توانید یک لیست را با استفاده از `args*` به تابع ارسال کنید و مقادیر داخل آن را به صورت جداگانه دریافت کنید.

3. جمع کردن اعداد مختلف ارسال شده به تابع با استفاده از `args*\`

یکی از کاربردهای متداول `args*`، جمع کردن تعداد متغیر آرگومان‌ها است. به راحتی می‌توانید با استفاده از یک حلقه یا توابع built-in مانند `()sum` تمامی اعداد ارسال شده را جمع کنید.

مثال:

```
def sum_numbers(*args):
    return sum(args)

فراخوانی تابع با آرگومان‌های مختلف #
result = sum_numbers(1, 2, 3, 4, 5)
print(result)
```

در اینجا:

- از تابع `()sum` برای جمع کردن تمامی مقادیر موجود در `args` استفاده می‌شود.
- تابع `sum_numbers` با دریافت هر تعداد عدد، آن‌ها را جمع می‌کند.

خروجی برنامه:

```
15
```

4. مزایای استفاده از `args*\`:

- **انعطاف‌پذیری:** با استفاده از `args*` می‌توانید توابعی بنویسید که تعداد متغیری از آرگومان‌ها را پذیرش کنند.
- **ساده‌سازی کد:** به جای اینکه تعداد زیادی پارامتر برای تابع تعریف کنید، می‌توانید فقط از یک پارامتر با نام `args*` استفاده کنید و تعداد نامحدودی از آرگومان‌ها را به آن ارسال کنید.

- **پشتیبانی از لیست‌ها و تاپل‌ها:** با استفاده از `args*` می‌توانید به راحتی لیست‌ها یا تاپل‌ها را به عنوان آرگومان به تابع ارسال کنید.

نکات:

- `args*` همیشه باید در آخرین پارامترهای تابع قرار بگیرد. اگر بخواهید هم از پارامترهای نامشخص و هم از پارامترهای مشخص استفاده کنید، باید پارامترهای مشخص ابتدا آمده و سپس از `args*` برای پارامترهای متغیر استفاده کنید.
- به جز `args*`، از `kwargs**` نیز می‌توان برای پذیرش آرگومان‌های کلیدی (keyword arguments) استفاده کرد که مشابه `args*` اما به صورت دیکشنری عمل می‌کند.

در این بخش شما یاد گرفتید که چگونه از `args*` برای پذیرش تعداد نامحدودی از آرگومان‌ها استفاده کنید. همچنین نمونه‌هایی از کاربرد آن در جمع کردن اعداد و ارسال لیست به تابع را مشاهده کردید. این تکنیک‌ها به شما کمک می‌کنند تا توابعی بنویسید که انعطاف‌پذیرتر و مقیاس‌پذیرتر باشند.

==========

استفاده از `kwargs**`

در پایتون، `kwargs**` به شما این امکان را می‌دهد که تعداد نامحدودی از آرگومان‌ها را به صورت کلید-مقدار (keyword arguments) به یک تابع ارسال کنید. به عبارت دیگر، با استفاده از `kwargs**` می‌توانید آرگومان‌های ورودی را به صورت جفت‌های کلید-مقدار دریافت کنید. در این حالت، آرگومان‌های ارسالی به صورت یک دیکشنری به تابع منتقل می‌شوند.

1. معرفی `kwargs**\` برای ارسال آرگومان‌های کلید-مقدار به تابع

پارامتر `kwargs**` به تابع این امکان را می‌دهد که یک دیکشنری از کلیدها و مقادیر دریافت کند. برخلاف `args*` که آرگومان‌ها را به صورت یک تاپل می‌گیرد، `kwargs**` کلیدها را به عنوان نام‌گذاری برای مقادیر و در قالب یک دیکشنری دریافت می‌کند.

چطور کار می‌کند؟
وقتی از `kwargs**` استفاده می‌کنید، تمامی آرگومان‌های کلید-مقدار ارسال شده به تابع به صورت یک دیکشنری به نام `kwargs` ذخیره می‌شوند که کلیدها نام آرگومان‌ها و مقادیر مربوط به آنها هستند.

2. توضیح نحوه استفاده از `kwargs**\` برای دریافت تعداد نامحدود از آرگومان‌های کلید-مقدار

برای استفاده از `kwargs**`، باید در تعریف تابع از دو ستاره (`**`) قبل از نام پارامتر استفاده کنید. این پارامتر به صورت یک دیکشنری عمل کرده و هر جفت کلید-مقدار که به تابع ارسال می‌شود در آن ذخیره خواهد شد.

مثال:

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

# فراخوانی تابع با آرگومان‌های کلید-مقدار مختلف
print_person_details(name="Ali", age=30, occupation="Engineer")
```

در اینجا:

- تابع `print_person_details` با استفاده از `kwargs**` تمام آرگومان‌های کلید-مقدار را دریافت می‌کند.
- داخل تابع، از حلقه `for` و متد `()items.` برای دسترسی به کلیدها و مقادیر دیکشنری استفاده شده است.
- می‌توانیم برای هر فرد، اطلاعاتی مانند نام، سن، شغل و هر جزئیات دیگری را به صورت کلید-مقدار ارسال کنیم.

خروجی برنامه:

```
name: Ali
age: 30
occupation: Engineer
```

3. مثال از ایجاد یک تابع که جزئیات مربوط به یک شخص را دریافت کرده و چاپ کند

در این مثال، تابعی نوشته‌ایم که جزئیات مربوط به یک شخص شامل نام، سن و شغل را دریافت کرده و آن‌ها را چاپ می‌کند. این اطلاعات به صورت کلید-مقدار ارسال می‌شوند.

مثال:

```
def describe_person(**kwargs):
    name = kwargs.get('name', 'Unknown')
    age = kwargs.get('age', 'Not provided')
    occupation = kwargs.get('occupation', 'Not specified')

    print(f"Name: {name}")
    print(f"Age: {age}")
    print(f"Occupation: {occupation}")

# فراخوانی تابع با آرگومان•های کلید-مقدار
describe_person(name="Sara", age=25, occupation="Teacher")
```

در اینجا:

- از متد `()get.` برای استخراج مقادیر از دیکشنری `kwargs` استفاده شده است.
- اگر کلیدی موجود نباشد، مقدار پیش‌فرض به عنوان مقدار برگشتی در نظر گرفته می‌شود (مثلاً `'Unknown'` برای `name`).

خروجی برنامه:

```
Name: Sara
Age: 25
Occupation: Teacher
```

4. مزایای استفاده از `kwargs**\`:

- **انعطاف‌پذیری بالا:** می‌توانید تعداد نامحدودی از آرگومان‌ها را به تابع ارسال کنید بدون اینکه نیاز به مشخص کردن دقیق تعداد پارامترها در تعریف تابع داشته باشید.
- **قابلیت استفاده از نام‌های معنادار:** به جای ارسال مقادیر بدون نام، با استفاده از `kwargs**` می‌توانید از نام‌های معنادار (کلیدها) برای مقادیر استفاده کنید که کد را خوانا و قابل فهم‌تر می‌کند.

- **پشتیبانی از مقادیر پیش‌فرض:** می‌توانید مقادیر پیش‌فرض برای هر کلید تعیین کنید، که در صورتی که کلیدی ارسال نشد، از آن استفاده شود.

نکات:

- `kwargs**` باید در آخرین پارامترهای تابع قرار بگیرد. اگر هم از `args*` و هم از `kwargs**` استفاده می‌کنید، باید `args*` اول و `kwargs**` در انتها قرار گیرد.
- می‌توانید همزمان از پارامترهای معمولی، `args*` و `kwargs**` در یک تابع استفاده کنید.

با استفاده از `kwargs**` می‌توانید توابعی بنویسید که به راحتی تعداد نامحدودی از آرگومان‌های کلید-مقدار را دریافت کنند و با استفاده از آن‌ها، اطلاعات متنوعی را پردازش یا نمایش دهید. این تکنیک به شما انعطاف‌پذیری بالایی در نوشتن کد می‌دهد.