পাইথনে হাতেখড়ি

আহমাদ আল-সাজিদ

পাইথনের ওপর ধারাবাহিক লেখায় এ পর্বে ফাংশন সম্পর্কে আলোচনা করা হয়েছে। কোনো সমস্যা সমাধানের জন্য আমরা যখন প্রোগ্রাম লিখি, তখন সেগুলোকে ছোট ছোট ভাগে ভাগ করে নিতে পারি। এই ছোট সমস্যাগুলো সমাধান করার জন্য প্রয়োজনীয় কোড আলাদা করে রাখার জন্য ফাংশন লিখতে পারি। এতে প্রোগ্রামিং যেমন সহজ হয়, তেমনি প্রোগ্রামারের জন্য কোড পড়ে বুঝতেও সুবিধা হয়।

পাইথনে ফাংশন ডিফাইন করার নির্দিষ্ট নিয়ম আছে। এর জন্য ফাংশনের নামের আগে def লিখতে হবে এবং ফাংশন নামের পরে ()-এর মধ্যে প্যারামিটার দিতে হবে। তবে প্যারামিটার না দিলেও () অবশ্যই দিতে হবে। এরপর কোলন (:) চিহ্ন দিতে হবে। ফ্যাক্টরিয়ালের মান বের করার জন্য যদি আমরা একটি ফাংশন লিখি তা হবে–

```
def fact(n):
    result=1
    while n>0:
        result = result * n
        n = n - 1
    print(result)
fact(5)
```

এই প্রোগ্রামটি রান করলে আমরা উত্তর পাব ১২০। এখানে আমরা একটি প্যারামিটার n পাস করেছি। প্যারামিটার পাস না করেও আমরা ফাংশন লিখতে পারি।

```
def prt():
    for i in range(3):
        print(‘computer jagat’)
prt()
```

এই প্রোগ্রামটি রান করলে আমরা উত্তর দেখতে পারব

```
>>>
computer jagat
computer jagat
computer jagat
>>>
```

অর্থাৎ আমরা ফাংশনের মধ্যে কিছু কাজের কথা বলে দিচ্ছি। এরপরই আমরা যখন সেই ফাংশনটিকে কল করব, তখনই সেই কাজগুলো সম্পন্ন হবে। ফাংশনের প্যারামিটারের মান যখন ফাংশনটিকে কল করে, তখন পাস করে দেয়। কিন্তু চাইলে আমরা প্যারামিটারগুলোর মান নির্দিষ্ট করে দিতে পারি। ফলে যদি আমরা ফাংশনে প্যারামিটার পাস না করি, তাহলে ফাংশনটি ওই নির্দিষ্ট মানটিকে নিয়ে কাজ করবে। যেমন– এখানে আমরা একটি ফাংশন নিলাম, যার প্যারামিটার দুটি a এবং b। ফাংশনটি a-কে b দিয়ে ভাগ করলে ভাগফল দেখাবে। এখনে b-এর ডিফল্ট ভ্যালু হিসেবে ১০ দেয়া আছে। অর্থাৎ যদি আমরা কল করার সময় b-এর কোনো ভ্যালু না দেই, তাহলে সে ১০ দিয়ে ভাগ করবে। আর b-এর ভ্যালু ফাংশনে দিয়ে দিলে সে ১০-এর বদলে ওই সংখ্যা দিয়ে ভাগ করবে।

```
def div(a , b = 10):
    print(a/b)
div (15, 4)
div (15)
```

প্রোগ্রামটি রান করলে আমরা রেজাল্ট পাবো

```
>>>
3.75
1.5
>>>
```

ফাংশন কল করার জন্য কিওয়ার্ড আর্গুমেন্ট (keyword arguments) ব্যবহার করা যায় kwarg = value ফর্মে। উদাহরণ হিসেবে নিচের ফাংশনটি দেখা যেতে পারে–

```
def parrot(voltage, state=’a stiff’, action=’voom’, type=’Norwegian Blue’):
        print(“— This parrot wouldn’t”, action, end=’ ‘)
        print(“if you put”, voltage, “volts through it.”)
        print(“— Lovely plumage, the”, type)
        print(“— It’s”, state, “!”)
```

এই ফাংশনটি কল করতে গেলে আমাদের অবশ্যই voltage-এর ভ্যালু পাস করতে হবে, আর বাকি তিনটি প্যারামিটারের মান না দিলেও চলবে। আমরা বেশ কয়েকভাবে এই ফাংশনটিকে কল করতে পারি। যেমন–

parrot(1000) # এখানে আমরা একটি পজিশনাল আর্গুমেন্ট দিচ্ছি।

parrot(voltage=1000) # এখানে আমরা একটি কিওয়ার্ড আর্গুমেন্ট দিচ্ছি।

parrot(voltage=1000000, action=’VOOOOOM’) # এখানে আমরা দুটি কিওয়ার্ড আর্গুমেন্ট দিচ্ছি।

parrot(action=’VOOOOOM’, voltage=1000000) # এখানে আমরা দুটি কিওয়ার্ড আর্গুমেন্ট দিচ্ছি।

parrot(‘a million’, ‘bereft of life’, ‘jump’) # এখানে আমরা তিনটি পজিশনাল আর্গুমেন্ট দিচ্ছি।

parrot(‘a thousand’, state=’pushing up the daisies’) # এখানে আমরা একটি পজিশনাল আর্গুমেন্ট এবং একটি কিওয়ার্ড আর্গুমেন্ট দিচ্ছি।

কিন্তু আমরা যদি অন্যভাবে এই ফাংশন কল করার চেষ্টা করি, তাহলে এরর দেখাবে। যেমন–

parrot() # কারণ এখানে কোনো প্যারামিটার দেয়া হয়নি।

parrot(voltage=5.0, ‘dead’) # কিওয়ার্ড আর্গুমেন্টের পরে নন-কিওয়ার্ড আর্গুমেন্ট দিলে এরর দেখাবে।

parrot(110, voltage=220) # একই আর্গুমেন্টের দুটি মান দিলে।

parrot(actor=’John Cleese’) # অজানা কিওয়ার্ড আর্গুমেন্ট দিলে।

তাই ফাংশন কল করার ক্ষেত্রে সতর্কতা বজায় রাখতে হবে।

যদি ফাংশনের প্যারামিটারের ক্ষেত্রে *name এবং **keywards এই ধরনের আর্গুমেন্ট দেখা যায়, তাহলে প্রথমটির ক্ষেত্রে সব ধরনের ভ্যালু পজিশনাল আর্গুমেন্ট এবং **keywards ডিকশনারি টাইপ ডাটা রিসিভ করবে। উদাহরণ হিসেবে নিচের প্রোগ্রামটি রান করে দেখতে পারি।

```
def function(*names, **types):
    for name in names:
        print(name,end=’ ‘)
    print()
    keys = sorted(types.keys())
    for key in keys:
        print(key,” : “,types[key])
```

এখন আমরা ফাংশনটিকে কল করব এভাবে–

```
function(‘cat’,’dog’,’horse’, cat=’kitty’, dog=’max’,horse=’star’)
```

এর ফলে আমরা যে রেজাল্ট দেখতে পারব, তা এমন হবে–

```
>>>
cat dog horse
cat  :  kitty
dog  :  max
horse  :  star
>>>
```

সবশেষে ফাংশন কল করার একটি অপ্রচলিত উপায় হচ্ছে অবাধ (arbitrary) সংখ্যক প্যারামিটার পাস করার উপায় রাখা। এই পদ্ধতিতে প্যারামিটারগুলো টিউপল (tuple) হিসেবে পাস হয়।

```
def concat(*args, sep=’/’):
return sep.join(args)
```

এই ফাংশনটিকে আমরা দুইভাবে কল করতে পারি–

```
print(concat(‘a’,’b’,’c’))
print(concat(‘a’,’b’,’c’,sep=’-’))
```

এতে আমরা দুই ধরনের উত্তর পাব–

```
>>>
a/b/c
a-b-c
>>>
```

কিছু ক্ষেত্রে এর উল্টো হতে পারে। অর্থাৎ ডাটা টিউপল বা লিস্ট হিসেবে আছে। কিন্তু ফাংশনে এদের আলাদা করা দরকার হতে পারে। যেমন– range() ফাংশনের ক্ষেত্রে দুটি প্যারামিটার শুরু এবং শেষ ব্যবহার করা যায়। সে ক্ষেত্রে ফাংশন কলটি *- অপারেটরের সাহায্য নিয়ে লেখা যেতে পারে, যার কাজ হবে লিস্ট বা টিউপল থেকে ভ্যালুগুলো আলাদা করা। উদাহরণ হিসেবে নিচের প্রোগ্রামটি রান করানো যাক–

```
>>> args = [3, 6]
>>> list(range(*args))
[3, 4, 5]
>>>
```

কোন ফাংশন কী কাজে লাগছে এর জন্য ডকুমেন্টেশন করতে হয়। অর্থাৎ ফাংশনের বিভিন্ন বর্ণনা থাকে, যাতে অন্য কেউ প্রোগ্রামটি দেখলে বুঝতে পারে। এর জন্য ফাংশন নামের নিচে তিনটি (‘ বা “) চিহ্ন দিয়ে ডকুমেন্টেশন শুরু করা হয় এবং শেষ হলে আবার তিনটি (‘ বা “) চিহ্ন দিয়ে ডকুমেন্টেশন শেষ করা হয়। এরপর চাইলেই আমরা আমাদের ফাংশনের ডকুমেন্টেশন পড়ে দেখতে পারি প্রিন্টের মাধ্যমে।

```
def function():
    “”“do nothing, but document it.
        it really dose nothing
    “”“
    pass
```

নিচের প্রিন্ট কমান্ডটি দিলে আমরা ডকুমেন্টেশনটি দেখতে পারব–

```
print(function.__doc__)
```

এর ফলে আমরা কন্সোলে দেখতে পারব–

```
>>>
do nothing, but document it.
    it really dose nothing
>>>
```

ফাংশন সম্পর্কে আরও বিস্তারিত জানতে পাইথনের অফিসিয়াল ডকুমেন্টেশন দেখা যেতে পারে কজ

ফিডব্যাক : ahmadalsajid@gmail.com