**HDB\_HW1**

**تمرین اول درس پایگاه داده پیشرفته**

**شکیلا طرهانی**

**سوال 1:** دریافت یک عدد به عنوان ورودی و سپس استخراج ارقام زوج از آن و نمایش آنها به صورت جدا شده با کاراکتر \* است.

**توضیحات برنامه:**

تابع extract\_even\_digits:

* 1. این تابع یک عدد را به عنوان ورودی می‌گیرد و یک رشته حاوی ارقام زوج جدا شده با \* را برمی‌گرداند.
  2. حلقه while: تا زمانی که عدد ورودی بزرگتر از صفر باشد تکرار می‌شود.
  3. استخراج رقم یکان: با استفاده از عملگر % 10 رقم یکان عدد استخراج می‌شود.
  4. بررسی زوج بودن: اگر رقم استخراج شده زوج باشد، به لیست even\_digits اضافه می‌شود.
  5. تقسیم صحیح: با استفاده از عملگر // 10، رقم یکان از عدد اصلی حذف می‌شود.
  6. برعکس کردن و اتصال ارقام: لیست ارقام زوج برعکس می‌شود تا ارقام به ترتیب صحیح نمایش داده شوند.
  7. بازگرداندن نتیجه: در نهایت، عناصر لیست even\_digits با استفاده از \*.join` به یک رشته تبدیل شده و برعکس می‌شود تا ترتیب ارقام در خروجی صحیح باشد.

گرفتن ورودی و چاپ نتیجه:

* 1. از کاربر خواسته می‌شود یک عدد وارد کند.
  2. تابع extract\_even\_digits فراخوانی شده و نتیجه آن در متغیر result ذخیره می‌شود.
  3. مقدار result چاپ می‌شود.

**سوال 2:** نوشتن برنامه‌ای برای محاسبه عبارت ریاضی داده شده برای 500 جمله. این عبارت شامل فاکتوریل اعداد و عملیات جمع، تفریق و تقسیم است.

**توضیحات برنامه:**

1. from sympy import factorial:

این خط کتابخانه SymPy را ایمپورت می‌کند و تابع factorial را از آن وارد می‌کند. این تابع برای محاسبه فاکتوریل اعداد استفاده می‌شود.

1. def calculate\_term(n, numerator, denominator)::

این خط یک تابع به نام calculate\_term تعریف می‌کند که سه ورودی می‌گیرد:

n: شماره جمله

Numerator: صورت کسر

Denominator: مخرج کسر

1. sign = (-1) \*\* (n-1):

این خط علامت جمله را محاسبه می‌کند. اگر n زوج باشد، علامت منفی و اگر فرد باشد، علامت مثبت می‌شود. این به دلیل الگوی تناوبی مثبت و منفی در عبارت اصلی است.

1. return sign \* factorial(numerator) / denominator:

این خط مقدار نهایی جمله را محاسبه و باز می‌گرداند. ابتدا علامت جمله با فاکتوریل صورت ضرب می‌شود و سپس بر مخرج تقسیم می‌شود.

1. def main()::

این خط تابع اصلی برنامه را تعریف می‌کند.

1. result = 0:

در ابتدای تابع main، یک متغیر به نام result با مقدار اولیه صفر تعریف می‌شود. این متغیر برای جمع کردن نتایج هر جمله استفاده می‌شود.

1. for i in range(1, 501)::

این حلقه for برای تکرار 500 بار اجرا می‌شود. در هر تکرار، متغیر i یک عدد از 1 تا 500 را به خود می‌گیرد که نشان‌دهنده شماره جمله است.

1. numerator = 2 \* i + 1:

در هر تکرار، صورت کسر با استفاده از فرمول 2\*i+1 محاسبه می‌شود که همیشه یک عدد فرد است.

1. denominator = i + 9 if i % 2 == 0 else -i + 10:

این خط مخرج کسر را بر اساس الگوی مشخص شده محاسبه می‌کند. اگر i زوج باشد، مخرج برابر با i+9 و اگر فرد باشد، مخرج برابر با -i+10 خواهد بود.

1. result += calculate\_term(i, numerator, denominator):

در هر تکرار، نتیجه محاسبه یک جمله به متغیر result اضافه می‌شود.

1. print("نتیجه نهایی:", result):

پس از پایان حلقه، مقدار نهایی result که مجموع تمام جملات است، چاپ می‌شود.

1. if \_\_name\_\_ == "\_\_main\_\_"::

این خط اطمینان حاصل می‌کند که تابع main تنها زمانی اجرا می‌شود که این فایل به عنوان یک اسکریپت اصلی اجرا شود و نه به عنوان یک ماژول وارد شود.

**سوال 3:** برنامه‌ای به زبان پایتون بنویس که چاپ تمام اعداد چهار رقمی (بین ۱۰۰۰ تا ۹۹۹۹) آورده شده است که در آن جمع دو رقم اخر با ضرب دو رقم اول برابر باشد.

**توضیحات برنامه:**

def check\_number(num)::

این خط یک تابع با نام check\_number تعریف می‌کند که یک عدد صحیح به عنوان ورودی می‌گیرد.

first\_digit = num // 1000:

با استفاده از تقسیم صحیح (//)، رقم هزارگان عدد استخراج می‌شود.

second\_digit = (num // 100) % 10:

ابتدا عدد بر 100 تقسیم می‌شود تا رقم‌های هزارگان و صدگان حذف شوند. سپس با استفاده از عملگر مدول (%)، رقم صدگان استخراج می‌شود.

third\_digit = (num // 10) % 10:

به همین ترتیب، رقم دهگان استخراج می‌شود.

fourth\_digit = num % 10:

رقم یکان با استفاده از عملگر مدول (%) استخراج می‌شود.

return (third\_digit + fourth\_digit) == (first\_digit \* second\_digit):

این خط شرط مسئله را بررسی می‌کند و مقدار True یا False را برمی‌گرداند. اگر مجموع رقم‌های دهگان و یکان برابر با حاصل‌ضرب رقم‌های هزارگان و صدگان باشد، تابع مقدار True را برمی‌گرداند.

results = []:

یک لیست خالی به نام results تعریف می‌شود تا اعداد جواب در آن ذخیره شوند..

for num in range(1000, 10000)::

یک حلقه for برای بررسی همه اعداد از 1000 تا 9999 تعریف می‌شود.

if check\_number(num)::

اگر عدد فعلی شرط مسئله را برآورده کند، دستورات داخل این شرط اجرا می‌شوند.

results.append(num):

عدد فعلی به لیست results اضافه می‌شود.

print(", ".join(map(str, results))):

عناصر لیست results به رشته تبدیل شده و با کاما از هم جدا می‌شوند و در نهایت چاپ می‌شوند.

**سوال 4:** برنامه ای بنویسید تا تمام اعداد 3 رقمی (بین 100 تا 999) را چاپ کند که ارقام فرد ندارد 0 را به عنوان رقم زوج در نظر بگیرید.

**توضیحات برنامه :**

def has\_only\_even\_digits(num)::

این تابع بررسی می‌کند که آیا همه ارقام عدد ورودی زوج هستند یا خیر.

for digit in str(num)::

این حلقه for برای هر رقم عدد ورودی اجرا می‌شود. در هر تکرار، متغیر digit یک رقم از عدد را به صورت یک رشته دریافت می‌کند.

if int(digit) % 2 != 0::

این شرط بررسی می‌کند که آیا رقم فعلی (که به صورت یک رشته است) وقتی به عدد تبدیل شود، بر 2 بخش‌پذیر نیست (یعنی فرد است).

return False:

اگر رقمی فرد پیدا شود، تابع بلافاصله مقدار False را برمی‌گرداند، زیرا شرط مسئله برقرار نیست.

return True:

اگر حلقه for به پایان برسد و هیچ رقم فردی پیدا نشود، یعنی همه ارقام زوج هستند و تابع مقدار True را برمی‌گرداند.

results = []:

یک لیست خالی به نام results تعریف می‌شود تا اعداد جواب در آن ذخیره شوند.

for num in range(100, 1000)::

یک حلقه for برای بررسی همه اعداد از 100 تا 999 تعریف می‌شود.

if has\_only\_even\_digits(num)::

اگر همه ارقام عدد فعلی زوج باشند، دستورات داخل این شرط اجرا می‌شوند.

results.append(num):

عدد فعلی به لیست results اضافه می‌شود.

print(", ".join(map(str, results))):

عناصر لیست results به رشته تبدیل شده و با کاما از هم جدا می‌شوند و در نهایت چاپ می‌شوند.

**سوال 5:** برنامه ای بنویسید تا از کاربر n عدد بگیرد و الگوی زیر را چاپ کند.

**توضیحات برنامه :**

دو حلقه تو در تو ایجاد می‌شود:

حلقه بیرونی برای کنترل شماره سطر است.

حلقه درونی برای چاپ اعداد در هر سطر استفاده می‌شود.

در هر تکرار حلقه درونی، حاصل‌ضرب شماره سطر در یک شمارنده (که از 1 شروع می‌شود و تا شماره سطر افزایش می‌یابد) محاسبه و چاپ می‌شود.

تابع print\_pattern:

این تابع یک عدد صحیح n را به عنوان ورودی دریافت می‌کند و الگو را چاپ می‌کند.

حلقه بیرونی با متغیر i از 1 تا n تکرار می‌شود که هر تکرار یک سطر جدید را نشان می‌دهد.

حلقه درونی با متغیر j از 1 تا i تکرار می‌شود و در هر تکرار، حاصل‌ضرب i\*j چاپ می‌شود.

end=" "باعث می‌شود که اعداد در همان خط چاپ شوند و با یک فاصله از هم جدا شوند.

print()یک خط جدید ایجاد می‌کند تا به سطر بعدی برویم.

چاپ الگو:

تابع print\_pattern با مقدار n فراخوانی می‌شود تا الگو چاپ شود.

**سوال 6:** برنامه ای بنویسید تا n و سپس n عدد از کاربر دریافت کند و حداکثر، حداقل، میانگین و انحراف معیار را محاسبه کند.

**توضیحات برنامه :**

تابع calculate\_statistics:

این تابع یک لیست از اعداد را دریافت می‌کند و بزرگ‌ترین، کوچک‌ترین، میانگین و انحراف استاندارد آن را محاسبه می‌کند.

از توابع آماده پایتون مانند max, min, sum و math.sqrt برای محاسبات استفاده می‌شود.

محاسبه و نمایش نتایج:

تابع calculate\_statistics برای محاسبه آماره‌ها فراخوانی می‌شود.

**سوال 7:** برای سوال 6، 4 تابع بنویسید. 5 عدد را از کاربر اصلی دریافت کنید. ارسال کنید .اعداد برای توابع Max1، Min1، Ave1 و STD1. Max1 n عدد می دهد و حداکثر برگردند. Min1 n عدد و حداقل بازگشت را می دهد. Ave1 و STD1 n عدد را می دهد و میانگین و انحراف استاندارد را به تنهایی چاپ می کند . بدنه و هیچ مقداری را بر نمی گرداند.

**توضیحات برنامه :**

تابع get\_numbers: از کاربر تعداد اعداد را گرفته و آن‌ها را در یک لیست ذخیره می‌کند.

تابع max1: بزرگ‌ترین عدد در لیست را پیدا کرده و برمی‌گرداند.

تابع min1: کوچک‌ترین عدد در لیست را پیدا کرده و برمی‌گرداند.

تابع ave1: میانگین اعداد را محاسبه کرده و چاپ می‌کند.

تابع std1: انحراف استاندارد اعداد را محاسبه کرده و چاپ می‌کند.

توابع ave1 و std1 فقط نتایج را چاپ می‌کنند و هیچ مقداری برنمی‌گردانند، همانطور که در سوال درخواست شده بود.

**سوال 8:** یک عدد با 5 رقم اصلی بدهید. اگر عدد 5 رقمی نباشد،مکرراً از کاربر بخواهید یک شماره معتبر وارد کند. شماره F1 را ارسال کنید تابعی برای یافتن و برگرداندن حداکثر رقم عدد. در بعدی ، حداکثر رقم و عدد ورودی تابع F2 را برای حذف ارسال کنید حداکثر رقم از عدد و آن را برگردانید. در نهایت خروجی نهایی را چاپ کنید

**توضیحات برنامه :**

تابع get\_5\_digit\_number: تا زمانی که کاربر یک عدد 5 رقمی وارد کند، از او درخواست تکرار می‌کند.

تابع find\_max\_digit: با استفاده از یک حلقه، ارقام عدد را بررسی کرده و بزرگترین آن‌ها را پیدا می‌کند.

تابع remove\_max\_digit: با استفاده از یک حلقه، ارقام عدد را بررسی کرده و ارقام غیر از حداکثر رقم را در یک عدد جدید قرار می‌دهد.

برای پیدا کردن ارقام یک عدد، از عملگرهای % (باقیمانده تقسیم) و // (تقسیم صحیح) استفاده شده است.

برای ساختن عدد جدید، از یک متغیر کمکی برای جمع کردن ارقام استفاده شده است.

**سوال 9:** متغیرهای جهانی و محلی: کدهای P1، P2، P3 و P4 را اجرا کرده و خروجی ها را گزارش دهید. دلایل خروجی آنها را با جزئیات توضیح دهید.

**توضیحات برنامه :**

متغیرهای محلی (Local Variables):

داخل یک تابع تعریف می‌شوند.

فقط در داخل همان تابع قابل دسترسی هستند.

وقتی تابع اجرا می‌شود، متغیر محلی ایجاد می‌شود و وقتی تابع به پایان می‌رسد، متغیر از بین می‌رود.

متغیرهای سراسری (Global Variables):

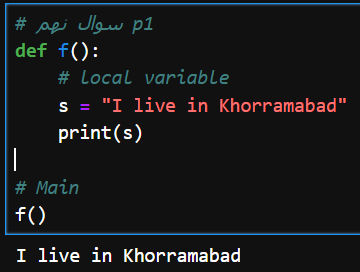
خارج از هر تابع تعریف می‌شوند.

در کل برنامه قابل دسترسی هستند.

تغییرات در یک متغیر سراسری در همه قسمت‌های برنامه اعمال می‌شود.



**P1:**



تعریف تابع f:

این خط یک تابع با نام f تعریف می‌کند. توابع بلوک‌هایی از کد هستند که برای انجام کارهای خاص تعریف می‌شوند و می‌توان آن‌ها را چندین بار فراخوانی کرد.

متغیر محلی s:

داخل تابع f، متغیری به نام s تعریف شده و مقدار "I live in Khorramabad" به آن اختصاص داده شده است.

این متغیر یک متغیر محلی است، به این معنی که فقط داخل تابع f قابل دسترسی است و در خارج از آن قابل مشاهده نیست.

چاپ مقدار s:

دستور print(s) مقدار متغیر s را در خروجی چاپ می‌کند.

فراخوانی تابع f:

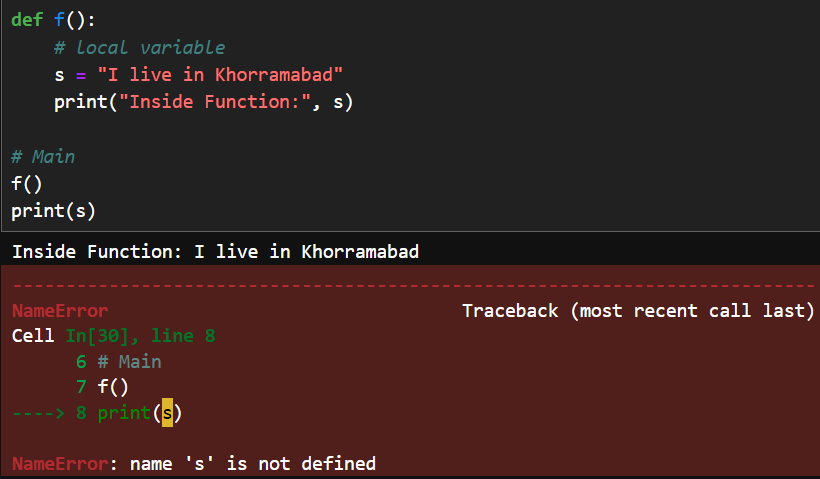
در قسمت اصلی برنامه، تابع f فراخوانی می‌شود. این باعث می‌شود که کدهای داخل تابع اجرا شوند.

نکات مهم:

متغیرهای محلی به حافظه اختصاص داده می‌شوند زمانی که تابع فراخوانی می‌شود و هنگامی که تابع به پایان می‌رسد، از حافظه آزاد می‌شوند.

استفاده از متغیرهای محلی باعث می‌شود که کدهای شما بهتر سازماندهی شده و خواناتر شوند. از ایجاد متغیرهای سراسری به جای متغیرهای محلی خودداری کنید، زیرا استفاده بیش از حد از متغیرهای سراسری می‌تواند منجر به مشکلات در برنامه‌های بزرگتر شود.

**P2:**



این کد شباهت زیادی به کد P1 دارد با این تفاوت که در انتهای برنامه سعی شده است تا متغیر s را خارج از تابع چاپ کنیم.

تعریف تابع f:

همانند کد P1، یک تابع با نام f تعریف می‌شود که یک متغیر محلی به نام s دارد و مقدار آن را چاپ می‌کند.

فراخوانی تابع f:

تابع f فراخوانی می‌شود و خروجی "Inside Function: I live in Khorramabad" را تولید می‌کند.

تلاش برای چاپ s خارج از تابع:

در خط آخر، سعی شده است تا مقدار متغیر s را خارج از تابع چاپ کنیم. اما این خط باعث ایجاد خطا می‌شود.

چرا خطا رخ می‌دهد؟

دامنه متغیرهای محلی: متغیر s یک متغیر محلی است و فقط داخل تابع f قابل دسترسی است. هنگامی که اجرای تابع f به پایان می‌رسد، متغیر s از حافظه حذف می‌شود و دیگر قابل دسترسی نیست.

خطای NameError: تلاش برای دسترسی به متغیری که تعریف نشده است، باعث ایجاد خطای NameError می‌شود.

خروجی نهایی:

خروجی این کد تنها خط "Inside Function: I live in Khorramabad" خواهد بود که مربوط به چاپ داخل تابع است. خط بعدی که سعی در چاپ s خارج از تابع دارد، باعث ایجاد خطا می‌شود.

نتیجه‌گیری:

این مثال به وضوح نشان می‌دهد که متغیرهای محلی تنها در محدوده تابعی که تعریف شده‌اند قابل دسترسی هستند و تلاش برای دسترسی به آن‌ها خارج از آن تابع باعث ایجاد خطا می‌شود. این مفهوم بسیار مهم در برنامه‌نویسی است و باید به خوبی درک شود تا از بروز خطاهای غیرمنتظره جلوگیری شود.

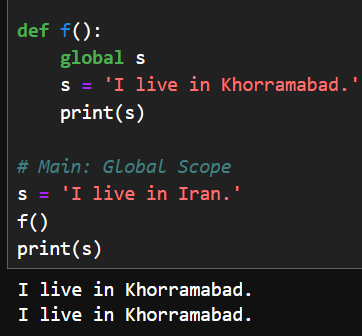
نکات کلیدی:

متغیرهای محلی فقط در داخل تابعی که تعریف شده‌اند قابل دسترسی هستند.

تلاش برای دسترسی به متغیر محلی خارج از تابع باعث ایجاد خطای NameError می‌شود.

استفاده از متغیرهای محلی باعث می‌شود که کدهای شما بهتر سازماندهی شده و خواناتر شوند.

**P3:**



این کد به بررسی نحوه استفاده از متغیرهای سراسری و محلی در پایتون می‌پردازد.

تعریف تابع f:

یک تابع به نام f تعریف می‌شود. در ابتدای تابع، از کلمه کلیدی global s استفاده شده است. این به پایتون می‌گوید که متغیر s که در داخل تابع استفاده می‌شود، همان متغیری است که در خارج از تابع نیز تعریف شده و یک متغیر سراسری است.

مقدار متغیر s به "I live in Khorramabad." تغییر می‌کند.

مقدار جدید s چاپ می‌شود.

بخش اصلی برنامه:

در ابتدا، یک متغیر سراسری به نام s تعریف شده و مقدار "I live in Iran." به آن اختصاص داده می‌شود.

سپس، تابع f فراخوانی می‌شود.در نهایت، مقدار نهایی متغیر سراسری s دوباره چاپ می‌شود.

توضیح خروجی:

هنگامی که تابع f فراخوانی می‌شود، به دلیل استفاده از کلمه کلیدی global، مقدار متغیر سراسری s به "I live in Khorramabad." تغییر می‌کند. سپس، این مقدار جدید چاپ می‌شود.

پس از پایان اجرای تابع، مقدار نهایی متغیر سراسری s که در بخش اصلی برنامه تغییر کرده است، دوباره چاپ می‌شود که همان "I live in Khorramabad." است.

چرا خروجی به این صورت است؟

استفاده از global s باعث می‌شود که هر تغییری که در متغیر s داخل تابع ایجاد شود، بر روی متغیر سراسری s نیز تأثیر بگذارد.

بنابراین، هنگامی که مقدار s داخل تابع به "I live in Khorramabad." تغییر می‌کند، این تغییر در سراسر برنامه اعمال می‌شود.

نکات مهم:

استفاده از global به طور گسترده توصیه نمی‌شود، زیرا می‌تواند باعث ایجاد کدهای پیچیده و دشوار برای اشکال‌زدایی شود.

بهتر است از متغیرهای محلی استفاده کنید و فقط در مواقع ضروری از متغیرهای سراسری استفاده کنید.

استفاده بیش از حد از متغیرهای سراسری می‌تواند منجر به تداخل بین بخش‌های مختلف برنامه شود.

جمع‌بندی:

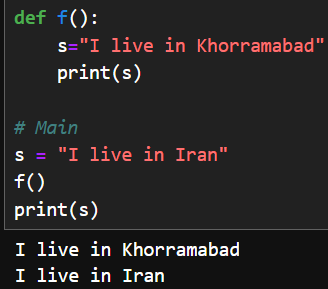
متغیرهای سراسری در کل برنامه قابل دسترسی هستند.

با استفاده از کلمه کلیدی global می‌توان به یک متغیر سراسری از داخل یک تابع دسترسی پیدا کرد و آن را تغییر داد.

استفاده بیش از حد از متغیرهای سراسری می‌تواند باعث پیچیدگی و مشکلات در برنامه شود.

بهتر است از متغیرهای محلی استفاده کنید و فقط در مواقع ضروری از متغیرهای سراسری استفاده کنید.

**P4:**



این کد شباهت زیادی به کدهای قبلی دارد، اما تفاوت کلیدی آن در نحوه استفاده از متغیر s داخل تابع است.

تعریف تابع f:

یک تابع به نام f تعریف می‌شود. داخل تابع، متغیری به نام s تعریف شده و مقدار "I live in Khorramabad." به آن اختصاص داده شده است. سپس، مقدار s چاپ می‌شود.

بخش اصلی برنامه:

در ابتدا، یک متغیر به نام s تعریف شده و مقدار "I live in Iran." به آن اختصاص داده می‌شود. سپس، تابع f فراخوانی می‌شود.

در نهایت، مقدار نهایی متغیر s دوباره چاپ می‌شود.

توضیح خروجی:

هنگامی که تابع f فراخوانی می‌شود، یک متغیر محلی به نام s داخل تابع ایجاد می‌شود و مقدار "I live in Khorramabad." به آن اختصاص داده می‌شود. سپس، این مقدار جدید چاپ می‌شود.

پس از پایان اجرای تابع، به بخش اصلی برنامه برمی‌گردیم. در اینجا، متغیر سراسری s که مقدار "I live in Iran." دارد، دوباره چاپ می‌شود.

چرا خروجی به این صورت است؟

در این کد، از کلمه کلیدی global استفاده نشده است. بنابراین، متغیر s داخل تابع f یک متغیر محلی است و تغییرات آن بر روی متغیر سراسری s تأثیری ندارد.

متغیر محلی s داخل تابع، یک کپی از متغیر سراسری s نیست. این دو متغیر کاملاً جدا از هم هستند و هر کدام فضای حافظه جداگانه‌ای را اشغال می‌کنند.

تفاوت با کد P3:

در کد P3، با استفاده از global s، متغیر s داخل تابع به متغیر سراسری اشاره می‌کرد و هر تغییری در آن، بر روی متغیر سراسری نیز تأثیر می‌گذاشت.

در این کد (P4)، متغیر s داخل تابع یک متغیر محلی است و تغییرات آن بر روی متغیر سراسری تأثیری ندارد.

نکات کلیدی:

اگر می‌خواهید تغییرات یک متغیر داخل تابع، بر روی متغیر سراسری تأثیر بگذارد، باید از کلمه کلیدی global استفاده کنید.

اگر از global استفاده نکنید، متغیر داخل تابع یک متغیر محلی است و تغییرات آن بر روی متغیر سراسری تأثیری ندارد.

اگر می‌خواهید در داخل یک تابع، مقدار یک متغیر سراسری را تغییر دهید، باید از global استفاده کنید.

اگر می‌خواهید یک متغیر جدید ایجاد کنید که فقط داخل تابع قابل دسترسی باشد، نیازی به استفاده از global نیست.

**سوال 10:** مطالعه توابع بازگشتی در پایتون .با جزئیات توضیح دهید که چگونه تابع بازگشتی زیر کار می کند؟

**توضیحات برنامه :**

این کد یک تابع بازگشتی به نام factorial را تعریف می‌کند که فاکتوریل یک عدد را محاسبه می‌کند.

تابع بازگشتی چیست؟ تابعی است که در تعریف خود، به خودش ارجاع می‌دهد. یعنی در داخل بدنه تابع، خود تابع دوباره فراخوانی می‌شود. این نوع توابع برای حل مسائلی که می‌توان آن‌ها را به زیرمسائل کوچکتر و مشابه تقسیم کرد، بسیار مفید هستند.

تعریف تابع:

def factorial(x): : این خط یک تابع با نام factorial تعریف می‌کند که یک پارامتر به نام x می‌گیرد.

شرط پایه:

if x == 1: : اگر مقدار x برابر با 1 باشد، تابع به صورت مستقیم عدد 1 را برمی‌گرداند. این شرط پایه است که به فراخوانی‌های بازگشتی پایان می‌دهد و از ایجاد یک حلقه بی‌نهایت جلوگیری می‌کند. زیرا فاکتوریل عدد 1 برابر با 1 است.

حالت بازگشتی:

else: : اگر x بزرگتر از 1 باشد، وارد این بخش می‌شویم.

return (x \* factorial(x-1)): : در این قسمت، تابع خود را با آرگومان x-1 فراخوانی می‌کند. این یعنی برای محاسبه فاکتوریل x، ابتدا فاکتوریل x-1 را محاسبه کرده و سپس نتیجه را در x ضرب می‌کند. این فرایند تا زمانی ادامه می‌یابد که به شرط پایه برسیم.

مثال: محاسبه فاکتوریل 4 وقتی factorial(4) را فراخوانی می‌کنیم، اتفاقات زیر رخ می‌دهد:

factorial(4) فراخوانی می‌شود.

از آنجایی که 4 بزرگ‌تر از 1 است، به بخش else می‌رود و 4 \* factorial(3) را محاسبه می‌کند.

factorial(3) فراخوانی می‌شود و همین روند تکرار می‌شود تا به factorial(1) برسیم.

factorial(1) به شرط پایه می‌رسد و 1 را برمی‌گرداند.

نتایج بازگشتی به ترتیب در هم ضرب می‌شوند:

factorial(2) = 2 \* 1 = 2

factorial(3) = 3 \* 2 = 6

factorial(4) = 4 \* 6 = 24

در نهایت، تابع مقدار 24 را برمی‌گرداند که همان فاکتوریل 4 است.

نحوه کار تابع بازگشتی: تابع بازگشتی به صورت تودرتو خود را فراخوانی می‌کند تا به یک جواب برسد. هر فراخوانی جدید، یک کپی از تابع با مقادیر متفاوت برای پارامتر ایجاد می‌کند. این فراخوانی‌ها تا زمانی ادامه می‌یابد که به شرط پایه برسند و سپس نتایج به صورت بازگشتی به فراخوانی‌های قبلی برگردانده می‌شوند.

مزایای توابع بازگشتی:

خوانایی کد: در بسیاری از موارد، توابع بازگشتی کد را خواناتر و قابل درک‌تر می‌کنند.

حل مسائل پیچیده: توابع بازگشتی برای حل مسائلی که به صورت بازگشتی تعریف می‌شوند، بسیار مناسب هستند.

معایب توابع بازگشتی:

مصرف حافظه: فراخوانی‌های بازگشتی متعدد می‌توانند باعث مصرف زیاد حافظه شوند.

-پیچیدگی در پیاده‌سازی: برخی از الگوریتم‌ها ممکن است با استفاده از حلقه‌های تکراری به صورت ساده‌تری پیاده‌سازی شوند.