جزوه آموزش مفاهیم اولیه پایتون

عملگرهای مقایسه‌ای (Comparison Operators)

- پایتون از عملگرهای مقایسه‌ای برای مقایسه مقادیر استفاده می‌کند. این عملگرها شامل == (برابر با)، < (کوچکتر از)، > (بزرگتر از)، <= (کوچکتر یا مساوی با) و >= (بزرگتر یا مساوی با) هستند.
- عملگر "مخالف برابر" به صورت != است.
- این عملگرها همیشه یک نتیجه بولی (True یا False) تولید می‌کنند.
- توجه داشته باشید که مقایسه بین انواع مختلف ممکن است نتیجه False برگرداند، به عنوان مثال: 4 == "4" برابر با False است.

دستورات شرطی (Conditional Statements)

- دستورات شرطی، که عمدتاً با استفاده از if استفاده می‌شوند، جریان کد را بر اساس شرایط خاص کنترل می‌کنند.
- یک دستور if نیاز به یک شرط دارد که نتیجه آن یا True یا False باشد. اگر شرط True باشد، بلوک کد داخل دستور if اجرا می‌شود.
- پایتون به جای آکولاد از تورفتگی (indentation) برای تعریف بلوک‌های کد استفاده می‌کند.
- دستور if می‌تواند با یک دستور else همراه باشد. در این صورت بلوک کد داخل else تنها زمانی اجرا می‌شود که شرط if برابر با False باشد.
- با استفاده از کلمه کلیدی elif می‌توان چندین شرط را به ترتیب بررسی کرد. اگر یک شرط if غلط باشد، شرط elif بررسی می‌شود. اگر یکی از شرایط True باشد، دیگر شرایط بررسی نمی‌شوند.
- دستورات if می‌توانند تو در تو (nested) باشند، اما بهتر است از تو در تو کردن بیش از حد خودداری کنید، زیرا کد غیرقابل خواندن می‌شود.
- در استفاده از if، elif و else، پس از اولین شرطی که True باشد، اجرای بلوک شرطی متوقف می‌شود.
- در صورت استفاده از چندین دستور مستقل if، هر شرط به طور جداگانه بررسی می‌شود.

عملگرهای منطقی (Logical Operators)

- پایتون از عملگرهای منطقی and، or و not برای ایجاد شرایط پیچیده استفاده می‌کند.
- عملگر and تنها زمانی True برمی‌گرداند که هر دو شرطی که به هم متصل می‌کند True باشند.
- عملگر or در صورتی که حداقل یکی از شرایط True باشد، مقدار True برمی‌گرداند.
- عملگر not مقدار بولی شرط را معکوس می‌کند: اگر شرط True باشد، مقدار False برمی‌گرداند و برعکس.

لیست‌ها (Lists)

- لیست‌ها برای ذخیره مجموعه‌ای از آیتم‌ها در یک ترتیب خاص استفاده می‌شوند.
- لیست‌ها با استفاده از براکت‌های مربع [] ایجاد می‌شوند.
- یک لیست می‌تواند شامل انواع داده‌ای مختلف در یک لیست باشد.
- لیست‌ها اجازه آیتم‌های تکراری را می‌دهند.
- ترتیب آیتم‌ها در لیست حفظ می‌شود.
- برای ایجاد یک لیست خالی می‌توان از [] یا list() استفاده کرد.
- یک رشته می‌تواند با استفاده از list() به لیستی از کاراکترها تبدیل شود.
- عدد صحیح نمی‌تواند مستقیماً با استفاده از list() به لیست تبدیل شود.

ایندکس‌گذاری لیست (List Indexing)

- آیتم‌های لیست با استفاده از ایندکس آن‌ها که از ۰ شروع می‌شود، دسترسی پیدا می‌کنند. به عنوان مثال، اولین آیتم دارای ایندکس ۰ است.
- ایندکس‌های منفی به عناصر انتهایی لیست دسترسی دارند، به عنوان مثال a[-1] به آخرین عنصر دسترسی پیدا می‌کند.
- دسترسی به ایندکسی که خارج از محدوده باشد باعث ایجاد خطای IndexError می‌شود.

متدهای لیست (List Methods)

- append(item): آیتمی را به انتهای لیست اضافه می‌کند.
- remove(item): اولین نمونه از آیتم مشخص شده را حذف می‌کند. اگر آیتم وجود نداشته باشد، خطای ValueError رخ می‌دهد.
- insert(index, item): آیتم را در ایندکس مشخص شده درج می‌کند و آیتم‌های دیگر را جا به جا می‌کند.
- pop(): آخرین عنصر لیست را حذف کرده و آن را برمی‌گرداند.
- pop(index): عنصر موجود در ایندکس مشخص شده را حذف کرده و برمی‌گرداند. اگر ایندکس خارج از محدوده باشد، خطای IndexError رخ می‌دهد.
- del list[index]: آیتم موجود در ایندکس مشخص شده را از لیست حذف می‌کند. اگر ایندکسی مشخص نشده باشد، کل لیست حذف می‌شود.
- این متدها مستقیماً لیست را تغییر می‌دهند و لیست جدیدی برنمی‌گردانند.

توابع داخلی برای لیست‌ها (Built-in Functions for Lists)

- len(list): تعداد آیتم‌های موجود در لیست را برمی‌گرداند.
- min(list): کوچک‌ترین آیتم موجود در لیست را برمی‌گرداند. اگر لیست دارای انواع مختلفی باشد که قابل مقایسه نیستند، خطا رخ می‌دهد.
- max(list): بزرگ‌ترین آیتم موجود در لیست را برمی‌گرداند. اگر لیست دارای انواع مختلفی باشد که قابل مقایسه نیستند، خطا رخ می‌دهد.
- sum(list): مجموع تمام آیتم‌های عددی موجود در لیست را برمی‌گرداند. این تابع نمی‌تواند لیستی با انواع مختلط را جمع بزند.
- sorted(list): لیستی مرتب شده جدیدی را برمی‌گرداند.
  - آرگومان reverse=True می‌تواند برای مرتب‌سازی نزولی استفاده شود.
- list.index(item): ایندکس اولین وقوع مقدار مشخص شده در لیست را برمی‌گرداند. اگر مقدار وجود نداشته باشد، خطای ValueError رخ می‌دهد.
- list.count(item): تعداد دفعات ظاهر شدن یک آیتم خاص در لیست را برمی‌گرداند.

حلقه‌ها (Loops)

- پایتون دارای دو نوع حلقه است: while و for.
- حلقه‌های while تا زمانی که یک شرط True باشد، اجرا می‌شوند. شرط قبل از هر بار اجرای حلقه بررسی می‌شود و مدیریت متغیرهای حلقه مهم است تا حلقه بی‌نهایت نشود.
- حلقه‌های for بر روی یک دنباله از آیتم‌ها (مانند لیست‌ها یا بازه‌ها) تکرار می‌کنند. نحو آن به صورت for item in sequence: است. متغیر حلقه item پس از پایان حلقه همچنان در دسترس است و مقدار نهایی آن آخرین آیتم دنباله خواهد بود.
- حلقه‌های for می‌توانند از طریق رشته‌ها تکرار کنند و هر کاراکتر به عنوان یک آیتم در نظر گرفته می‌شود.

دستورات کنترل حلقه (Loop Control Statements)

- continue: ادامه اجرای حلقه را متوقف کرده و به تکرار بعدی می‌رود.
- break: اجرای حلقه را به طور کامل متوقف می‌کند.

بازه (Range)

- تابع range() دنباله‌ای از اعداد را برای تکرار تولید می‌کند.
- range(stop): یک دنباله از اعداد از ۰ تا (اما شامل نشدن) مقدار stop ایجاد می‌کند.
- range(start, stop): یک دنباله از مقدار start تا (اما شامل نشدن) مقدار stop ایجاد می‌کند.

- range(start, stop, step): یک دنباله از مقدار start تا مقدار stop (شامل نشدن) با گام مشخص step ایجاد می‌کند. گام می‌تواند منفی باشد.
- range() یک لیست تولید نمی‌کند، بلکه اعداد را در صورت نیاز تولید می‌کند.
- می‌توانید از حلقه‌های for برای تکرار در یک بازه استفاده کنید.
- حلقه‌ها می‌توانند تو در تو باشند، به این معنا که می‌توانند شامل حلقه‌های دیگر باشند.

عملگر والروس (Walrus Operator)

- عملگر والروس := مقداری را به یک متغیر در یک عبارت اختصاص می‌دهد.
- این عملگر در حلقه‌های while برای اختصاص و بررسی شرایط به طور همزمان مفید است.
- عملگر والروس می‌تواند در دستورات if نیز استفاده شود.
- متغیری که با استفاده از عملگر والروس ایجاد شده است، پس از پایان حلقه با آخرین مقدار اختصاص داده شده خود در دسترس است.
- هنگام استفاده از عملگر والروس، بهتر است تخصیص مقدار را در پرانتز قرار دهید تا متغیر به درستی محدوده‌بندی شود.