جزوه کامل از مفاهیم مختلف برنامه‌نویسی در پایتون

1. طراحی توابع و مدیریت ورودی‌ها:

- **مسئله: ایجاد تابعی که تعداد متغیرهای ورودی را محاسبه کرده و مجموع آن‌ها را برگرداند.**
- **نکته: در مسائل واقعی، تمام جزئیات مشخص نیستند و برنامه‌نویسان باید تمام جنبه‌ها را در نظر بگیرند.**
- **تابع نباید مجموع مقادیر قبلی را ذخیره کند و باید با هر مجموعه ورودی جدید مجموع را محاسبه کند.**
- **روش‌هایی برای حل مسئله:**
  1. **استفاده از توابع آماده len() و sum().**
  2. **استفاده از حلقه برای شمارش عناصر و محاسبه مجموع.**
- **مدیریت آرگومان‌های کلیدی (kwargs) و موقعیتی (args).**
- **نیازی به دریافت ورودی جدید در داخل تابع نیست.**

2. سیستم‌های عددی و نمایش باینری:

- **داده‌ها در کامپیوتر با استفاده از سیستم باینری (مبنای 2) ذخیره می‌شوند.**
- **تبدیل مبنای 10 به باینری:**
  - **تقسیم مکرر بر 2 و ذخیره باقی‌مانده‌ها به صورت معکوس.**
  - **مقدار مکانی هر بیت برابر است با 2 به توان مکان آن بیت.**
  - **بیت‌های بااهمیت کم (LSB) و بااهمیت زیاد (MSB).**
- **فقط بیت LSB تعیین می‌کند که عدد باینری فرد یا زوج است.**

3. تبدیل باینری به دهدهی:

- **هر بیت در مقدار مکانی خود ضرب شده و مجموع حاصل به‌دست می‌آید.**
- **توابع پایتون:**
  - **bin() برای تبدیل عدد دهدهی به رشته باینری.**
  - **int()برای تبدیل رشته عددی در یک مبنای مشخص به عدد دهدهی.**
    - **مثال: int('1100100', 2).**
    - **اگر مبنا مشخص نشود، به صورت پیش‌فرض مبنای 10 در نظر گرفته می‌شود.**
    - **ورودی int() باید یک رشته باشد.**
- **پیشوندهای "0b" (باینری)، "0o" (هشت‌هشتی) و "0x" (شانزده‌شانزده) مبنای عدد را مشخص می‌کنند.**

4. سیستم‌های عددی هشت‌هشتی و شانزده‌شانزده:

- برای ساده‌سازی اعداد باینری طولانی استفاده می‌شوند.
- تبدیل باینری به هشت‌هشتی: گروه‌بندی بیت‌ها به دسته‌های سه‌تایی.
- تبدیل باینری به شانزده‌شانزده: گروه‌بندی بیت‌ها به دسته‌های چهارتایی (A-F برای مقادیر 10 تا 15).
- توابع پایتون:
  - oct() و hex() برای تبدیل به رشته‌های هشت‌هشتی و شانزده‌شانزده.
- تبدیل به مبنای 10 از سایر مبناها با استفاده از مجموع ضرب ارقام در مبنای به توان مکان رقم انجام می‌شود.

5. کاربردهای شانزده‌شانزده و کدگذاری کاراکترها:

- برای نمایش رنگ‌ها و آدرس‌های حافظه استفاده می‌شود.
- کدگذاری کاراکتر:
  - ASCII:
    - از یک بایت (8 بیت) برای هر کاراکتر استفاده می‌کند.
    - شامل حروف انگلیسی، اعداد و برخی نمادها.
    - توابع پایتون: ord() (کد ASCII) و chr() (تبدیل کد به کاراکتر).
  - UTF-8:
    - از 1 تا 4 بایت برای هر کاراکتر استفاده می‌کند.
    - قابلیت نمایش کاراکترهای بیشتری، از جمله ایموجی‌ها.
    - کاراکترهای ASCII در UTF-8 مشابه هستند.

6. مدیریت خطا با بلوک‌های Try-Except:

- بلوک try-except برای مدیریت خطاها در زمان اجرا.
- خطاهای مختلف:
  - SyntaxError، ModuleNotFoundError، ZeroDivisionError، ValueError، TypeError.
- Traceback: نشان می‌دهد خطا کجا رخ داده و چه پیامی دارد.
- ساختار:
  - کدی که ممکن است خطا ایجاد کند در بلوک try قرار می‌گیرد.
  - بلوک except برای مدیریت خطاها.
  - بلوک else در صورت عدم وقوع خطا اجرا می‌شود.
  - بلوک finally برای تمیزکاری و بستن منابع، بدون توجه به وقوع خطا اجرا می‌شود.

- مدیریت خطاهای خاص با انواع مختلف except:
  - مثال: except ZeroDivisionError :، except TypeError:.
- تنها یک بلوک finally در هر ساختار try-except می‌تواند وجود داشته باشد.

7. ایجاد خطا با raise:

- استفاده از raise برای تولید خطاهای خاص.
- زمانی استفاده می‌شود که ورودی‌ها نامعتبر هستند یا عملیات نمی‌تواند تکمیل شود.

8. بررسی‌های برنامه با assert:

- کلمه کلیدی assert برای بررسی وضعیت برنامه.
- اگر شرطی نادرست باشد، AssertionError تولید می‌شود.
- مناسب برای توسعه و اشکال‌زدایی، اما نه برای کد نهایی.

9. مدیریت فایل‌ها:

- باز کردن فایل با open():
  - حالت‌ها: 'r' (خواندن)، 'w' (نوشتن و بازنویسی)، 'a' (افزودن).
  - خروجی: یک شی فایل برای خواندن یا نوشتن.
  - کدگذاری پیش‌فرض: UTF-8.
- خواندن فایل:
  - متد read() تمام فایل یا تعداد مشخصی کاراکتر را می‌خواند.
  - متد readline() یک خط را می‌خواند.
  - اشاره‌گر فایل موقعیت فعلی خواندن/نوشتن را مشخص می‌کند.
  - متد seek() برای جابجایی اشاره‌گر فایل.
- بستن فایل:
  - متد close() منابع را آزاد کرده و تغییرات را ذخیره می‌کند.
  - استفاده از with open() برای بستن خودکار فایل.
- نوشتن در فایل:
  - کاراکتر (\n) newline برای افزودن خطوط جدید.
  - حالت 'w' فایل را بازنویسی می‌کند.
  - حالت 'a' به انتهای فایل اضافه می‌کند.
  - حالت 'x' فایل جدید ایجاد کرده و اگر فایل وجود داشته باشد، FileExistsError تولید می‌کند.
- حذف فایل یا پوشه:

- **تابع os.remove(filepath) برای حذف فایل.**
- **تابع os.rmdir(folderpath) برای حذف پوشه.**
- **تابع os.path.exists(filepath) برای بررسی وجود فایل قبل از حذف.**