Wprowadzenie do programowania w Pythonie

Podstawowe moduły

Uwaga: Wszystkie programy powinny wywoływać funkcje! Dodatkowo należy użyć konstrukcji:

```
if __name__ == 'main:'
```

1 Moduł `os`

1. **Tworzenie i zarządzanie plikami w katalogu:**
   Napisz krótki program, który:
   - Sprawdza bieżący katalog roboczy (funkcja `os.getcwd()`).
   - Tworzy folder o nazwie `raporty`, jeśli jeszcze nie istnieje.
   - Wchodzi do folderu `raporty`.
   - Tworzy w pętli trzy pliki tekstowe o nazwach `raport1.txt`, `raport2.txt`, `raport3.txt`.
   - Wypisuje zawartość bieżącego katalogu w terminalu.

   Wskazówki:

```
import os

# 1. sprawdz i wyswietl sciezke
print("Aktualny katalog:", os.getcwd())

# 2. stworz folder raporty, jesli nie istnieje
# 3. przejdz do folderu
# 4. utworz pliki tekstowe w petli
# 5. wyswietl zawartosc katalogu
```

2. **Zliczanie rozmiarów plików:**
   Rozszerz powyższy program tak, aby:
   - W każdym z plików `raport{n}.txt` zapisać krótki tekst (np. kilka słów).
   - Sprawdzić rozmiar każdego z tych plików (w bajtach) funkcją `os.path.getsize()`.
   - Wyświetlić łączną sumę rozmiarów tych plików.

2 Moduł `sys`

1. **Argumenty wiersza poleceń:**
   Napisz program, który:
   - Wypisuje wszystkie argumenty przekazane w wierszu poleceń.
   - Sprawdza, czy został podany drugi argument (np. nazwa pliku lub ciąg znaków).
   - Jeśli tak – wyświetla stosowny komunikat, w przeciwnym razie zwraca informację o braku argumentu.

   Wskazówka: Możesz uruchomić program np.:

```
python skrypt.py arg1 arg2
```

2. **Dostosowywanie ścieżki wyszukiwania modułów:**
   Napisz program, który:
   - Wyświetli zawartość `sys.path`.
   - Doda tymczasowo do `sys.path` nowy katalog (np. `/home/user/moj_modul` - dla systemu Linux, w przypadku innego systemu dopasuj ścieżkę).
   - Ponownie wyświetli zawartość `sys.path` w celu porównania.

3 Moduł `math`

1. **Rozszerzone obliczenia geometryczne:**
   Napisz program, który:
   - Prosi użytkownika o promień `r`.
   - Oblicza i wyświetla pole koła oraz obwód koła, używając `math.pi`.
   - Oblicza i wyświetla objętość kuli o tym promieniu ($\frac{4}{3}\pi r^3$).
2. **Funkcje trygonometryczne i logarytmy:**
   - Napisz kod, który dla kolejnych kątów (np. 0, 30, 45, 60, 90 stopni) obliczy wartości `sin` i `cos` (nie zapomnij przeliczyć stopni na radiany!).
   - Wyświetl wyniki w przyjaznej formie, np.:

```
Kat (stopnie) | sin(kat) | cos(kat)
----------------------------------
...           | ...      | ...
```

4 Moduł `numpy`

Uwaga: Przed rozpoczęciem ćwiczeń upewnij się, że masz zainstalowany pakiet `numpy` (np. `pip install numpy`).

1. **Losowe dane i statystyki:**
   Napisz program, który:
   - Importuje `numpy` i generuje tablicę 100 losowych liczb z przedziału $[0, 1)$ (np. `np.random.rand(100)`).
   - Oblicza wartość średnią, odchylenie standardowe (`np.std`), wartość maksymalną i minimalną w tej tablicy.
   - Wyświetla podsumowanie w terminalu.
2. **Operacje na macierzach:**
   - Stwórz dwie macierze 3×3 (np. zawierające liczby całkowite) przy użyciu `np.array`.
   - Oblicz iloczyn macierzowy (`A @ B`) i porównaj go z iloczynem elementowym (`A * B`).
   - Dla uzyskanego wyniku (`A @ B`) wypisz sumę wszystkich elementów oraz sumy wzdłuż każdej kolumny.
3. **Zapisywanie i wczytywanie danych:**
   - Wygeneruj tablicę 2D, np. 5×4, z losowymi wartościami (np. `np.random.randint`).
   - Zapisz ją do pliku tekstowego (np. `np.savetxt("moje_dane.txt", tablica, fmt="%d")`).
   - Wczytaj ponownie dane z pliku (np. `dane = np.loadtxt("moje_dane.txt")`).
   - Porównaj, czy wczytana tablica jest identyczna z oryginałem (np. przy użyciu `np.array_equal`).

5 Moduł `time`

1. **Odmierzanie czasu wykonania:**
   Napisz krótki skrypt, który:
   - Odczytuje bieżący czas (sekundy od epoki) na początku.
   - Wykonuje pewną pętlę (np. `for i in range(10000000): pass`).
   - Odczytuje bieżący czas po zakończeniu pętli.
   - Wyświetla informację o tym, ile sekund trwało wykonanie pętli.
2. **Dokładne czekanie (sleep) i formatowanie:**
   - Napisz program, który odlicza 5 sekund, wypisując co sekundę aktualną godzinę i minutę (użyj `time.strftime` dla lepszego formatu, np. `"%H:%M:%S"`).
   - Po zakończeniu odliczania wyświetla stosowny komunikat (np. `"Gotowe!"`).

6 Moduł `datetime`

1. **Porównywanie dat:**
   - Napisz program, który pobiera od użytkownika dwie daty w formacie `YYYY-MM-DD`.
   - Tworzy obiekty `date` z tych danych (`datetime.date(rok, miesiac, dzien)`).
   - Wyświetla, która z dat jest wcześniejsza, a która późniejsza.
   - Oblicza i wyświetla, ile dni dzieli te daty.
2. **Różnica czasu w sekundach:**
   - Napisz program, który pyta o dwie daty i godziny (np. `"2025-04-16 12:30"`).
   - Konwertuje łańcuchy znaków na obiekty `datetime.datetime`.
   - Oblicza różnicę między nimi i wyświetla wynik w sekundach (np. `delta.total_seconds()`).
3. **Dodawanie przedziałów czasowych:**
   - Skorzystaj z `datetime.timedelta`, aby dodać np. 7 dni do bieżącej daty.
   - Wyświetl w terminalu nową datę w wygodnym formacie (np. `%d-%m-%Y`).

Wskazówki do zadań:

- Staraj się łączyć funkcjonalności z różnych modułów — np. moduł `os` do stworzenia plików i moduł `time` do zbadania, ile zajmuje to czasu.
- Zwróć uwagę na dobre praktyki kodowania: dzielenie programu na funkcje, używanie czytelnych nazw zmiennych, obsługa błędów itp.

Zadania opracował: Krzysztof Kotysz