Grupa I sprawdzian – Python
środa, 29 listopada 2023

1. **Każde zadanie powinno być wykonane w osobnym pliku.**
2. **Skrypty będące rozwiązaniami zadań umieść w katalogu** `Imie_Nazwisko_NrIndeksu` **(słowa** `Imie`**,** `Nazwisko` **i** `NrIndeksu` **zastąp swoim imieniem i nazwiskiem zapisanym bez polskich liter, oraz numerem indeksu).**
3. **Katalog spakuj do formatu .tar lub .zip, i prześlij mailem na adres** `Maciej.Buczynski@fuw.edu.pl`**.**

Zad. 1. (3 pkt) Stwórz skrypt `zadanie1.py`:

1. Zdefiniuj funkcję `collatz` z jednym parametrem `n`, która zwróci ilość kroków z problemu Collatza, w jakich `n` dojdzie do 1, tzn. ilość przekształceń postaci
$$c_{n+1} = \begin{cases} \frac{1}{2}c_n & \text{gdy } c_n \text{ jest parzysta} \\ 3c_n + 1 & \text{gdy } c_n \text{ jest nieparzysta} \end{cases}$$
2. Użyj zdefiniowanej funkcji, żeby znaleźć długość największego ciągu kroków dla liczb początkowych z przedziału 1 do 1000.

Wskazówka: przykładowe wartości funkcji collatz(1)=0, collatz(7)=16, collatz(27)=111.

Zad. 2. (3 pkt) Stwórz skrypt `zadanie2.py`:

1. Zdefiniuj funkcję `derivative` która będzie przyjmować tablicę `y` oraz liczbę `dx`. Funkcja powinna zwracać tablicę wartości pochodnych zgodnie ze wzorem:
$$\frac{df}{dx}(x) = \frac{f(x+dx) - f(x)}{dx}$$
Przyjmij, że punkty w tablicy `y` są równo oddalone od siebie o wartość `dx`.
2. Przetestuj funkcję `derivative` rysując poniższe wykresy oraz ich pochodne. W tym celu stwórz siatkę 4 rysunków (użyj np. `plt.subplots`), gdzie na każdym z nich będzie umieszczony inny wykres funkcji wraz z wykresem pochodnej:
   - $y = \sin(x)$
   - $y = x^2$
   - $y = \frac{1}{x^2+1}$
   - $y = e^x$
3. Dodaj tytuł do każdego z 4 rysunków.

Wskazówka: tablica wynikowa funkcji derivative będzie o 1 krótsza niż wejściowa. Przycięcie pierwszego/ostatniego elementu można zrobić np. poprzez y[1:] lub y[:-1].

Zad. 3. (4 pkt) Stwórz skrypt `zadanie3.py`:

1. Wczytaj dane (`x`, `y`, `y_err`) z pliku points2.dat
(lub ~mb459899/sprawdzian/python/points2.dat)
2. Dopasuj do punktów funkcję, która według Ciebie najlepiej opisuje zależność (używając `curve_fit` z modułu `scipy.optimize`).
3. Podczas dopasowania ustaw początkową wartość parametrów.
4. Narysuj na jednym wykresie punkty (niepołączone ze sobą!) ze słupkami błędów, oraz dopasowaną funkcję.
5. Wypisz na ekran wartości dopasowanych parametrów oraz ich niepewności (można je uzyskać wyciągając pierwiastek z diagonali macierzy kowariancji, przykładowo `np.sqrt(np.diag(cov))`).
6. Opisz osie i dodaj legendę.
7. Zapisz rysunek do pliku `fit.pdf`.