Programowanie I

Zajęcia nr 8

Rafał Masełek

27. kwietnia 2022r.

Zadanie 1. `pochodna` – Różnica centralna

Różnica centralna jest numerycznym sposobem na liczenie pochodnej funkcji f w punkcie x:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \tag{1}$$

gdzie h jest parametrem określającym dokładność liczenia pochodnej.
Napisz funkcję *pochodna*, która przyjmuje trzy parametry: funkcję do zróżniczkowania, wartość parametru h, oraz listę punktów, w których chcemy policzyć pochodną. Dodaj odpowiednie instrukcje do skryptu, które wywołają funkcję *pochodna* dla $y = x^2$, $h = 0.15$ oraz kilku dowolnie wybranych punktów na przedziale $[0, 1]$. Wywołanie ma się odbywać wyłącznie wtedy, gdy skrypt jest uruchamiany bezpośrednio. Gdy skrypt jest importowany jako moduł wewnątrz innego skryptu to zamiast wywołanie ma wyświetlić się komunikat informujący o zaimportowaniu modułu.
W programie nie wolno użyć żadnej pętli.

Zadanie 2. – Proste lambdy operujące na jednej zmiennej liczbowej

Używając funkcji lambda zaimplementuj następujące funkjonalności:

1. podnoszenie do kwadratu
2. dodawanie do 2
3. liczenie wartości wyrażenia $\sin \frac{\pi x}{9}$
4. zwracanie wartości prawda jeśli liczba jest większa bądź równa 5, w przeciwnym wypadku zwracanie fałszu
5. zwracanie 1 jeśli liczba jest podzielna przez 2, w przeciwnym razie 0
6. dzielenie przez 2 bez reszty
7. konwersja na typ float
8. zwracanie napisu "cyfra N" z liczbą podstawioną za N

Stwórz listę ze wszystkimi funkcjami lambda. Następnie używając pętli for iteruj po tej liście i dla każdej z funkcji lambda użyj funkcji *map*, żeby wywołać funkcję lambda dla wszystkich 10 cyfr. Wynik działania wypisz na standardowe wyjście.
W programie wolno użyć tylko jednej pętli.

Zadanie 3. – Proste lambdy operujące na napisach

Zadanie analogiczne do poprzedniego, ale tym razem operujemy na napisach. Funkcje do zaimplementowania:

1. zamień pierwszą literę w napisie na wielką
2. odwróć napis, tzn. "Napis" → "sipaN"
3. zamień spacje na znaki podkreślenia
4. usuń spacje i wstaw średniki "-" pomiędzy wszystkie pozostałe znaki

Wywołaj wszystkie funkcje lambda dla napisu "Ala ma kota, a Jacek ma psa.". Skorzystaj z funkcji map. W programie nie wolno użyć żadnej pętli.

Zadanie 4. – Więcej lambd

Napisz i przetestuj następujące funkcje lambda:

1. funkcja *pow* przyjmująca dwa argumenty: x oraz n i zwracająca x^n, gdzie podanie n jest opcjonalne, domyślnie jest to $n = 1$,
2. funkcja przyjmująca dwa argumenty x i y i zwracająca $\sqrt{x^2 + y^2}$; przetestuj jej działanie używając dwóch list i funkcji map,
3. funkcja przyjmującą dowolną liczbę argumentów i zwracającą ich średnią; nie używaj map ani pętli, przetestuj na dowolnym przykładzie

W programie nie wolno użyć żadnej pętli.

Zadanie 5. – Ułamki piętrowe

Dany jest ciąg zdefiniowany następująco:

$$a_1 = 1 \tag{2}$$

$$a_2 = \frac{1+2}{2} \tag{3}$$

$$a_3 = \frac{\frac{1+2}{2}+3}{3} \tag{4}$$

$$a_n = \frac{a_{n-1}+n}{n} \tag{5}$$

Napisz program, który przyjmuje liczbę k jako parametr uruchomienia, oblicza wartość a_k i wyświetla na standardowe wyjście. Program może mieć maksymalnie dwie linijki kodu (wskazówka: użyj *functools.reduce*). Pomiń sprawdzanie poprawności argumentu.
W programie nie wolno użyć żadnej pętli.

Zadanie 6. – Szyfr Cezara

Szyfr Cezara to prosta metoda szyfrowania, która polega na przesunięciu w alfabecie w sposób cykliczny wszystkich liter z szyfrowanej wiadomości o pewną stałą wartość zwaną kluczem, która jest liczbą całkowitą. Np. jeśli kluczem jest 1, to napis "azalia" przejdzie w "babmjb" ($z \to a$). Klucz może być większy niż liczba liter w alfabecie, odpowiada to jakby kilkukrotnemu "nawinięciu" alfabetu. Klucz może być również zerem (brak szyfrowania) lub ujemny (litery przesuwają się w drugą stronę).
Napisz program, który przyjmuje dwa parametry wywołania: klucz i napis do przesunięcia. Sprawdź liczbę argumentów. Sprawdź czy napis zawiera wielkie litery, jeśli tak to rzuć wyjątek i poproś użytkownika o podanie wyłącznie małych liter (cyfry i znaki specjalne są dozwolone, nie chcemy wielkich liter, żeby zrobić zadanie prostszym). Korzystając z *list comprehesion* albo funkcji lambda zaimplementuj w **jednej linijce** szyfrowanie wiadomości (chodzi o samo szyfrowanie). Wyświetl wynik działania na standardowe wyjście. Szyfrowane mają być jedynie litery, inne znaki mają pozostać bez zmiany. Program ma działać poprawnie dla dowolnej całkowitej wartości klucza. Podpowiedź: zapoznaj się z tabelą ASCII.

Zadanie 7. – Różności

1. Jednolinijkowy kod. Wypisz wszystkie liczby z przedziału $x \in [0, 10)$ dla których $\sin x > 0$.

2. Jednolinijkowy kod. Sprawdź czy w liście jest liczba podzielna przez 13 i wypisz True/False. Przetestuj dla dwóch różnych list, jednej zawierającej liczbę podzielną przez 13 i jednej bez.

3. Jednolinijkowy kod. Sprawdź czy wszystkie liczby w liście są parzyste i wypisz True/False. Przetestuj dla dwóch list, jednej tylko z parzystymi liczbami i jednej z liczbami obojga parzystości.

4. Dana jest "baza danych" pracowników zaimplementowana za pomocą słownika:

```
slownik = {
                    '1' : ('Jan', 'Kowalski', 4500),
                    '2' : ('Janina', 'Nowak',  5500),
                    '3' : ('Jacek', 'Morawiecki', 3500),
                     }
```

   Wypisz słownik, a następnie posortuj go malejąco po pensji pracowników (ściśle mówiąc zastąp go posortowaną wersją) i wypisz ponownie. Do sortowania uzyj funkcji *sorted* i funkcji lambda. Sortowanie da się zrealizować w jednej linijce.

   W programie nie wolno użyć żadnej pętli.