

## Technologie Informacyjne i Komunikacyjne R Python

**Zad. 1.** Poćwicz korzystanie z Pythona, jako podręcznego kalkulatora. Wykonaj następujące obliczenia:

- $2^{10}$
- $\sin \pi/3$
- $\log_{10} 10$
- reszta z dzielenia 17 przez 3.

*Wskazówka:* Przydatne może okazać się importowanie modułów `pylab` oraz `math`.

**Zad. 2.** Napisz program `date.py`, wypisujący na ekran aktualną datę i godzinę oraz UNIXtimestamp (liczba sekund liczona od początku epoki – January 1st, 1970 at UTC).

**Zad. 3.** W pliku `funkcje.py` zdefiniuj trzy funkcje: liczące pole koła, trójkąta i kwadratu. Wczytaj z klawiatury niezbędne dane i policz dla nich wszystkie trzy pola. Wynik wypisz na ekran.

**Zad. 4.** Napisz program `suma.py`, używający pętli `for` do liczenia sumy liczb całkowitych od 1 do 100.

**Zad. 5.** Napisz program `reszta.py`, wypisujący powitanie i informację co robi, a następnie proszący o wpisanie najpierw jednej, a potem drugiej liczby całkowitej i sprawdzający czy pierwsza dzieli się bez reszty przez drugą i wypisujący na ekran stosowny komunikat.

**Zad. 6.** Napisz program `pierwiastki.py` wyliczający pierwiastki równania kwadratowego. Program powinien wczytać ze standardowego wejścia współczynniki równania kwadratowego, a następnie różnie reagować w zależności od obliczonej delty.

**Zad. 7.** Napisz program `krotki.py`, w którym zdefiniujesz krotkę (`tuple`): `pocket = ('pen', 'pencil', 'keys')`. Sprawdź, czy masz w kieszeni klucze, wypisz na ekran odpowiednią wiadomość. Następnie wypisz na ekran, ile masz przedmiotów w kieszeni i jakie dwa umieściłeś w niej, jako ostatnie. Zdefiniuj nową krotkę, zawierającą jeden element `add = ('coins',)` (pamiętaj o przecinku). Spróbuj dodać nową krotkę do istniejącej i wypisz zawartość kieszeni na ekran. Następnie umieść wszystkie elementy w większej kieszeni.

**Zad. 8.** Napisz program `slovníki.py`, w którym zdefiniujesz słownik opisujący samochód. Jako klucze słownika podaj markę, model i rok produkcji (np. `brand`: 'Ford', `model`: 'Mustang', `year`: 1967). Następnie utwórz następny słownik ze specyfikacją techniczną samochodu (np. `engine`: 'Shelby', `engine_model`: 'GT500', `engine_power_KM`: '405') i połącz oba słowniki w słownik `moj_samochod`, wykorzystując metodę `update`. Operując na słowniku `moj_samochod`, wyświetl komunikat zawierający informacje o marce, modelu samochodu oraz producencie i modelu silnika swojego samochodu. Zdefiniuj słownik, opisujący swój drugi samochód (np. `brand`: 'Tesla', `model`: 'Blue Star', `year`: 2019, `engine`: 'battery', `engine_model`: 'lithium-ion 2170', `engine_power_KM`: 462). A następnie utwórz słownik nadrzędny, z kluczami `pierwszy`, który będzie odnosił się do pierwszego samochodu (tu Ford Mustang), i `drugi`, który będzie odnosił się do drugiego samochodu (tu Tesla Blue Star). Wypisz wszystkie swoje samochody.

**Zad. 9.** Napisz program `gradebook.py`, w którym utworzysz listę `classes = ['math', 'physics']`. Następnie dodaj do tej listy dwa dodatkowe przedmioty: `computer_science` oraz `electronics` za pomocą operatora dodawania. Posortuj listę alfabetycznie. Utwórz następną listę `grades = [88, 95, 90, 75]`, zawierającą oceny z poszczególnych przedmiotów. Złącz (*złącz*) obie listy i wynik tej operacji zapisz do listy-dzienniczka `gradebook`. Do "dzienniczka" wpisz także ocenę `['python', 100]` za pomocą funkcji `append` i wypisz zawartość listy `gradebook` na ekran.

**Zad. 10.** Napisz program `listy.py`, w którym utworzysz dwie listy `a` i `b`, po 3 liczby każda. Następnie w nowej liście `greater` zapiszesz wynik sprawdzenia, czy elementy listy `a` są większe niż elementy listy `b`. Listę `greater` wypisz na ekran. **Uwaga!** Celem ćwiczenia jest wykonanie operacji logicznej na całych listach, a nie element po elemencie.

**Zad. 11.** Napisz program `pascal.py`, wypisujący w konsoli trójkąt Pascala o rozmiarze zadanym przez użytkownika.

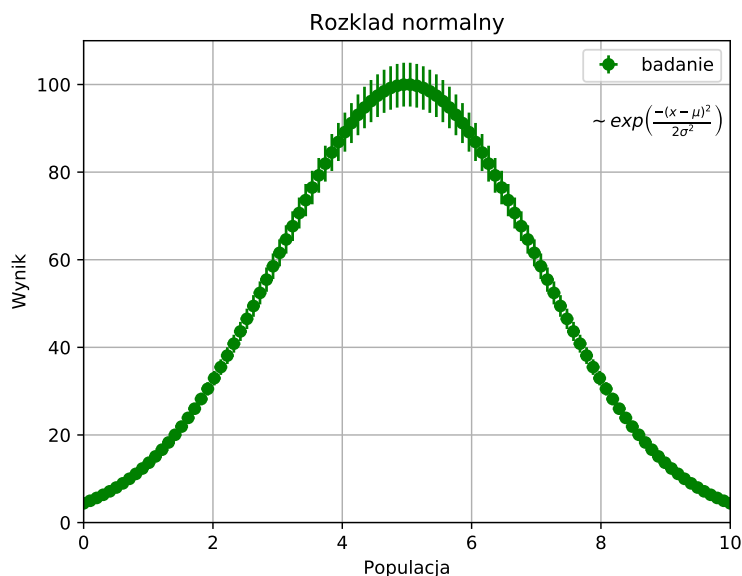
**Zad. 12.** Utwórz dwie macierze jednowymiarowe: macierz `A`, wypełnioną parzystymi liczbami całkowitymi z przedziału `[1,10]` i macierz `B`, wypełnioną ujemnymi, parzystymi liczbami całkowitymi z przedziału `[-10,-1]`. Uwaga! Pierwszą macierz utwórz korzystając z listy, tj. utwórz listę, którą następnie wczytasz do tablicy z biblioteki `numpy`. Do utworzenia drugiej macierzy wykorzystaj z funkcji `arange` z biblioteki `numpy`. Następnie utwórz macierz `C`, której element o indeksie `i` jest iloczynem elementów o indeksie `i` w tablicach `A` oraz `B`. Wypisz na ekran liczby z macierzy `A`, `B` i `C`.

**Zad. 13.** Napisz program `dzialania_na_macierzach.py`, w którym utworzysz listę zawierającą 100 liczb pseudolosowych z przedziału od 1 do 100 (włącznie), podlegających rozkładowi jednorodnemu. Następnie wczytaj listę to tablicy i wyznacz średnią wartość i odchylenie standardowe wygenerowanych liczb oraz sprawdź ile procent liczb jest większych niż 50.

**Zad. 14.** Napisz program `hiostogram.py`, który wygeneruje 1000 liczb pseudolosowych w przedziale od 0 do 10 podlegających rozkładowi trójkątnemu o medianie równej 3. Następnie utwórz i narysuj histogram zawierający wygenerowane liczby.

**Zad. 15.** Napisz program `sinus.py`, w którym korzystając z funkcji `linspace` z biblioteki `numpy` utworzysz wektor `X` zawierający 10 punktów w przedziale  $(0; 2\pi)$  oraz wektor `Y`, zawierający wartości  $\sin(X)$ . Narysuj wykres `X-Y`. Następnie sprawdź, jak wykres będzie wyglądał, jeśli zwiększysz liczbę punktów w wektorze `X` do 100 elementów.

**Zad. 16.** W pliku `bell_curve.dat` zostały zapisane cztery kolumny liczb w formacie: `x`, `y`, błąd `x`, błąd `y`. Wczytaj dane z pliku i narysuj punkty z błędami (funkcja `errorbar` z biblioteki `matplotlib`). Opisz osie. Zrób legendę. Dodaj tekst napisany w LaTeX'u opisujący krzywą dzwonową, a rysunek zapisz do pliku `bell.pdf`.



**Zad. 17.** Napisz program `bell_fit.py`, w którym, korzystając z danych z poprzedniego zadania wczytanych do macierzy, wykonasz następujące polecenia:

- Dopasuj do punktów funkcję  $f(x) = a \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
- Przy dopasowaniu zwróć uwagę na to, żeby użyć błędów punktów (opcje `sigma` oraz `absolute_sigma = True` funkcji `curve_fit`).
- Zadbaj o ustawienie początkowych wartości parametrów.
- Narysuj punkty z błędami (kolor zielony, duże kropki).
- Narysuj dopasowaną krzywą (kolor czerwony).
- Wypisz na ekran wartości dopasowanych parametrów z opisem i błędami.
- Zapisz rysunek do pliku `bell_fit.pdf`.

**Zad. 18.** Dla dopasowanej funkcji i danych z poprzedniego zadania policz wartość  $\chi^2/ndf$  gdzie:

$$\chi^2 = \sum_{i=1}^n \frac{(y_i - f(x_i; a_1, \dots, a_l))^2}{\sigma_i^2}$$

a  $ndf$  (liczba stopni swobody) to liczba punktów pomiarowych pomniejszona o liczbę dopasowywanych parametrów funkcji. Symbole  $a_i$  oznaczają parametry funkcji.

**Zad. 19.** W pliku `okres_wahadla_55x216.txt` znajdują się wyniki pomiarów okresu tego samego wahadła matematycznego (w sekundach), wykonane przez 55 studentów na pracowni. Każda linijka to pomiar wykonany przez jednego studenta. Napisz program `read_and_spit.py`, który wczyta (funkcja `loadtxt`) te dane do macierzy (użyj pakietu `numpy`), a następnie, korzystając z funkcji `savetxt` stworzy 55 plików o nazwach: `wahadlo_x.txt` ( $x \in [1, 55]$ ), zawierających zapisane w jednej kolumnie punkty pomiarowe pojedynczego studenta. Liczby powinny być zapisane w następującym formacie: na pięciu polach, z dokładnością do dwóch liczb po kropce.

**Zad. 20.** W programie `wahadlo.py`, wczytaj dane dla dowolnego pliku z poprzedniego zadania. Zrób z niego trzy histogramy: 1) krotności ( $n_i$ ), 2) częstości ( $\frac{n_i}{N}$ ) i 3) gęstości prawdopodobieństwa ( $f_i = \frac{n_i}{N\Delta_i}$ ), gdzie  $N$  jest całkowitą liczbą pomiarów, a  $\Delta$  szerokością przedziału w histogramie. Sprawdź, czy dla histogramu gęstości prawdopodobieństwa spełniona jest zależność:  $\sum_i^k f_j \Delta = 1$ , gdzie  $k$  jest liczbą binów histogramu.

*Wskazówki:*

- a by utworzyć trzy niezależne rysunki na trzech niezależnych panelach, lub jednym panelem podzielonym na trzy rysunki warto skorzystać ze środowiska `pyplot.figure`.
- użyj macierzy z pakietu `numpy` i polecenia `hist` z pakietu `matplotlib`; polecenie `hist` w wersji podstawowej: `n,bins,p=pyplot.hist(data)`; sprawdź co zwraca funkcja `hist` z pakietu `matplotlib`,
- do narysowania histogramów 2) i 3) możesz użyć wag (opcja `weights`) zdefiniowanych za pomocą polecenia `ones_like` z pakietu `numpy`
- opisz osie i podziel histogram na 10 przedziałów
- narysuj każdy z czterech histogramów w innym kolorze
- zapisz każdy z histogramów do innego pliku w formacie `pdf`.

**Zad. 21.** W programie `wahadla.py`, korzystając z plików z zadania 19 zrób dwa rysunki: jeden z nałożonymi na siebie histogramami krotności dla wszystkich 55 plików i drugi z histogramem wartości średnich. Opisz osie obu rysunków.

**Zad. 22.** Jeżeli  $n$  razy mierzymy pewną wielkość, np. okres wahadła matematycznego, to wielkość:  $R = \sum_i^N (x - \bar{x})^2$  jest miarą rozrzutu danych wokół wartości średniej, gdzie  $(\bar{x})$  jest średnim okresem wahadła. Stwórz wykres średniego  $R(n)$  dla  $n=2,3,4,6,8,12, 18$  i  $24$ . Użyj jednego z plików z zadania 19. W każdym z tych plików jest 216 pomiarów. Dlatego przykładowo dla  $n=18$  należy policzyć  $R$  dla 12 różnych grup pomiarów. (po 18 każdy), a następnie te wartości uśrednić. Opisz osie rysunku.

**Zad. 23.** Stwórz funkcję, która będzie przygotowywała rysunek jak w zadaniu poprzednim (bez `plt.show()`), dopasowywała zależność liniową do punktów, dodawała otrzymaną prostą do rysunku i wypisywała na ekran parametry dopasowania. Nazwa pliku powinna być argumentem funkcji. Następnie użyj tej funkcji do narysowania zależności  $R(n)$  na jednym rysunku dla pierwszych kilkunastu plików z zadania 19. W jakim punkcie krzywe się przecinają?

**Zad. 24.** Napisz program `klasa_complex.py`, w którym utworzysz klasę `ComplexNumber`. Wewnątrz klasy powinna znaleźć się funkcja inicjalizująca, tutaj tworząca dwa obiekty klasy – część rzeczywistą i urojoną liczby zespolonej. W klasie zawrzyj także metodę, pozwalającą na poprawne wyświetlanie liczby zespolonej (np.  $2 + 3i$ ), obliczanie modułu liczby zespolonej i jej argumentu. Następnie napisz metodę, która doda dwie liczby zespolone i wypisze

wynik, korzystając z wcześniej zdefiniowanej metody do wyświetlania liczb zespolonych.

**Zad. 25\*.** Napisz generator sudoku. Przykładowy wynik działania programu:

```
[. . . . 7 . . . 8]
[. . . . . 9 . . 4]
[. . . . . . . . .]
[. . . 3 . . . . .]
[. 2 . . . 4 . . 3]
[3 9 . 6 . . . 5 7]
[2 8 . . 4 6 . . .]
[. . . . 1 . . . .]
[. 1 . . 8 . 6 . .]
```