
	Politechnika Bydgoska im. J. J. Śniadeckich  <b>Wydział Telekomunikacji, Informatyki i Elektrotechniki</b>		
<b>Przedmiot</b>	Skryptowe języki programowania		
<b>Prowadzący</b>	mgr inż. Martyna Tarczewska		
<b>Temat</b>	<i>Więcej liczb</i>		
<b>Student</b>			
<b>Nr ćw.</b>	5	<b>Data wykonania</b>	
<b>Ocena</b>		<b>Data oddania spr.</b>	

## 1. Cel ćwiczenia

Celem ćwiczenia jest dalsze poznawanie sposobów zapisu i przetwarzania liczb, jakie oferuje język programowania *Python* oraz wykonanie zadań utrwalających przyswojoną wiedzę. **Funkcje i zmienne stworzone podczas zajęć powinny być odpowiednio otypowane z użyciem biblioteki *typing*. Kod powinien być napisany w języku angielskim.**

## 2. Informacje podstawowe

### 2.1. Liczby w Pythonie

Liczby (int) w Pythonie są zapisywane w formie obiektów. Domyślnie są wypisywane na ekranie jako liczby dziesiętne, jednak bardzo szybko można uzyskać dostęp do ich reprezentacji binarnej, ósemkowej i heksadecymalnej.

```
a = int(input("give me a dec number: "),10) #wczytanie w formacie dec.
print(bin(a))
print(oct(a))
print(hex(a))
```

### 2.2. Operacje bitowe

Python posiada również obsługę operacji bitowych, czyli wykonywanych na binarnych reprezentacjach liczb. Wszystkie operatory mogą też działać w miejscu, np. `&=` lub `^=`

```
x = 1
print(x << 2) # przesunięcie w lewo o 2 bity, wynik: 4
print(x >> 1) # przesunięcie w prawo o 1 bit, wynik: 0
print(x | 2) # OR, wynik: 3
print(x & 1) #AND, wynik: 1
print(x ^ 8) #XOR, wynik: 9

x ^= 5 #wynik x = 4
```

### 2.3. Moduł math

Na jednym z poprzednich laboratoriów stosowana była biblioteka math. Ma ona jest bardzo wiele funkcji, które warto znać i które czasem okażą się bezcenne.

```
a = float(input("give the number a: "))
b = int(input("give the number b: "))
print(math.trunc(a))
print(math.floor(a))
print(math.ceil(a))
print(math.lcm(int(a),b)) #najmniejsza wspólna wielokrotność
print(math.gcd(int(a),b)) #największy wspólny dzielnik
print(math.pow(int(a),b)) #potęgowanie
print(math.sqrt(a)) #pierwiastek kwadratowy
angle = 90
angleR = math.radians(angle) #konwersja na radiany
angle = math.degrees(angleR) #konwersja na stopnie
print(math.sin(angleR)) #sinus
print(math.cos(angleR)) #cosinus
```

### 2.4. Moduł random

Ten moduł umożliwia korzystanie z generatora liczb pseudolosowych.

```
import random
random.seed() #inicjalizacja generatora z czasem systemowym
random.random() #losowy float z zakresu <0, 1) 0.0 <= x < 1.0
random.uniform(2.5, 10.0) #losowy float z zakresu <2.5, 10>
random.randint(0, 10) #losowy int z zakresu <0, 10>
random.randrange(10) #losowy int z zakresu <0, 9>
random.randrange(0, 101, 2) # parzysty int z zakresu <0, 101)
random.choice(['win', 'lose', 'draw']) #losowy wybór elementu
deck = ['four', 'two', 'ace', 'three']
random.shuffle(deck) #przetasowanie elementów
print(deck)
```

### 3. Przebieg ćwiczenia

#### 3.1. Zadanie 1.

W oddzielnej funkcji wczytać podstawę systemu liczbowego (2, 8, 10 lub 16). Zabezpieczyć przed podaniem innych wartości. Następnie wczytać liczbę  $a$  (podaną w wybranym wcześniej formacie) i wypisać ją w postaci dwójkowej, ósemkowej, dziesiętnej i heksadecymalnej.

#### 3.2. Zadanie 2.

Napisać nową funkcję, która będzie pobierała dwie wartości. Pierwsza to liczba, druga to indeks bitu (zaczynamy numerowanie od 0, od prawej, od najmniej istotnego bitu). Funkcja ma zwracać wartość bitu z liczby na bicie oznaczonej indeksem. Ograniczyć program do liczb z zakresu  $\langle 0, 255 \rangle$ .

Np.

dla wartości (12, 0) otrzymamy 0, bo  $12(10) = 1100(2)$

dla wartości (12,2) otrzymamy 1, bo  $12(10) = 1100(2)$

#### 3.3. Zadanie 3.

Stworzyć nową funkcję, która wylosuje 3 liczby z zakresu  $\langle 3, 10 \rangle$ , sprawdzi czy z wylosowanych długości boków można zbudować trójkąt i jeśli tak, to obliczy jego pole.

#### 3.4. Zadanie 4.

Stworzyć funkcję, która będzie symulować rzuty monetą. Pobrać od użytkownika wartość orzeł/reszka i wylosować wynik. Zliczyć gry i wygrane. Ustalić sposób zakończenia gry (np. wczytanie znaku X).

#### 3.5. Zadanie 5.

Stworzyć funkcję, która będzie symulować grę w papier/kamień/nożyce. Pobrać od użytkownika wartość i wylosować wybór komputera. Zliczyć gry i wygrane. Ustalić sposób zakończenia gry (np. wczytanie znaku X).

#### 3.6. Zadanie 6.

Wyobrazić sobie (albo narysować) drabinę opartą o ścianę. Napisana funkcja ma pobierać dwie wartości: długość drabiny oraz kąt, pod jakim znajduje się drabina względem poziomu. Na podstawie podanych przez użytkownika wartości obliczyć wysokość na jaką sięga koniec drabiny. Wykorzystać moduł math.

#### 3.7. Zadanie 7.

Empirycznie (w nowej funkcji) sprawdzić, czy jedynka trygonometryczna sprawdza się również w Pythonie. W tym celu w pętli obliczyć wartość tego równania dla kątów z zakresu  $\langle 0^\circ, 90^\circ \rangle$ . Wykorzystać moduł math.

### 3.8. Zadanie 8.

Stworzyć nową funkcję realizującą kodowanie i dekodowanie szyfrem XOR (lub dwie oddzielne funkcje). W tym celu należy stworzyć tekst jawny i słowo szyfrowe (tej samej długości), np. „algorytm” i „kodykody”. Następnie należy w pętli pobierać kolejne znaki z napisów i wykonać operację XOR między nimi. Wynik tej operacji w postaci litery będzie znakiem tekstu niejawnego. Sprawdzić działanie napisanej funkcji poprzez jej ponowne użycie, tym razem za wejście podać tekst niejawny i to samo słowo kodowe.

**Wskazówka:** dostęp do i-tego elementu napisu uzyskujemy przez nawiasy klamrowe, funkcja `ord(char)` zwraca nam liczbę reprezentującą znak, funkcja `chr(int)` ma działanie odwrotne.

### 3.9. Zadanie 9.

Napisać funkcję w której użytkownik poda wykładnik potęgi **p**, a w wyniku otrzyma  $2^p$ .

Uwaga: użyć operatora przesunięcia bitowego.

### 3.10. Zadanie 10.

Przetestować w oddzielnej funkcji metody `math.trunc(a)`, `math.floor(a)` i `math.ceil(a)` dla podanej przez użytkownika liczby zmiennoprzecinkowej **a**. Sprawdzić w kodzie programu aktualną wersję Pythona, jeśli jest to 3.9 i wyżej, przetestować działanie metod `math.lcm(a,b)` i `math.gcd(a,b)`. Sprawdzić, czy w module `math` zawarta jest funkcja do obliczania wartości bezwzględnej.

### 3.11. Zadanie 11.

Wybrać jedną z utworzonych funkcji i przy pomocy biblioteki `pytest` utworzyć po trzy testy sprawdzające różne scenariusze działania wybranej funkcji. Wyniki testów zamieścić w sprawozdaniu.

## 4. Sprawozdanie

Sprawozdanie z laboratorium powinno zawierać:

- wypełnioną tabelę z początku instrukcji,
- kody programów będących rozwiązaniami wszystkich zadań wraz z komentarzami,
- demonstrację działania programu,
- odpowiedzi na pytania zawarte w zadaniach,
- wnioski.