Operatory

1. Operatory arytmetyczne

Operatory arytmetyczne takie jak +, -, *, /. Do oddzielenia działań wykorzystaj **nawiasy** ().Możesz ich używać zarówno z wyrażeniem **print** jak i w odniesieniu do **zmiennych**. Dodatkowo operator dodawania może być stosowany do łączenia ze sobą tekstów.

Operatory arytmetyczne	Przykład działania	Równoważny zapis działania
=	x = 2	x = 2
+=	x += 2	x = x + 2
-=	x -= 2	x = x – 2
*=	x *= 2	x = x * 2
/=	x /= 2	x = x / 2
//=	x //= 2	x = x // 2
%=	x %= 2	x = x % 2
**=	x **= 2	x = x ** 2

2. **Operatory przypisania**

```
x = 17
x += 2
x

# Przykład z potęgowaniem
x = 12
x **= 2
x

# Przykłąd dłuższej kalkulacji
x = (2+6) * 2 / 3
print(x)

# Przykład z wartością tekstową
x = []
x += "Hello World"
print(x)
```

Operacje porównania mogą zostać wykonane na wszystkich obiektach. Wszystkie mają ten sam priorytet (który jest wyższy od priorytetu operacji logicznych). Porównania mogą być jawnie łączone, na przykład zapis `x < y <= z` jest równoznaczny z `x < y and y <= z`, za wyjątkiem tego, że

wyrażenie *y* jest wyliczane tylko raz (ale w obu przypadkach *z* nie jest wyliczane w ogóle jeśli `x < y` okaże się fałszem).

Operator	Znaczenie operatora	Przykład wykorzystania
==	równy (zwraca „TRUE” jeśli obie wartości są takie same)	x == y
!=	nierówny (zwraca „TRUE” jeśli obie wartości są różne)	x != y
>	większy od (zwraca „TRUE” jeśli lewa wartość jest większa od prawej)	x > y
<	mniejszy od (zwraca „TRUE” jeśli lewa wartość jest mniejsza od prawej)	x < y
>=	większy lub równy (zwraca „TRUE” jeśli lewa wartość jest większa lub równa prawej)	x >= y
<=	mniejszy lub równy (zwraca „TRUE” jeśli lewa wartość jest mniejsza lub równa prawej)	x <= y

* `<>` i `!=` stanowią alternatywny zapis tego samego operatora. `!=` jest preferowaną formą; `<>` jest formą zanikającą.

Przykład zastosowania:

```
# Tworzymy dwie zmienne, które będziemy porównywać w kolejnych
działaniach

x = 3

y = 2


# Wyświetlamy obie wartości, by widzieć co z sobą porównujemy

print('x =', x)

print('y =', y)


# Wykorzystujemy operator „równy”, celem sprawdzenia, czy porównywane
wartości są równe

# W wyniku zostanie zwrócone „FALSE”, ponieważ porównywane wartości nie
są równe

# Wynik: Czy warunek x == y jest prawdziwy: False

print('Czy warunek x == y jest prawdziwy:', x == y)


# Wykorzystujemy operator „nierówny”, celem sprawdzenia, czy porównywane
wartości są różne

# W wyniku zostanie zwrócone „TRUE”, ponieważ porównywane wartości są
różne
```

```
# Wynik: Czy warunek x != y jest prawdziwy: True

print('Czy warunek x != y jest prawdziwy:', x != y)


# Wykorzystujemy operator „większy od”, celem sprawdzenia, czy lewa
wartość jest większa od prawej

# W wyniku zostanie zwrócone „TRUE”, ponieważ lewa wartość jest większa
od prawej

# Wynik: Czy warunek x > y jest prawdziwy: True

print('Czy warunek x > y jest prawdziwy:', x > y)


# Wykorzystujemy operator „mniejszy od”, celem sprawdzenia, czy lewa
wartość jest mniejsza od prawej

# W wyniku zostanie zwrócone „FALSE”, ponieważ lewa wartość jest mniejsza
od prawej

# Wynik: Czy warunek x < y jest prawdziwy: False

print('Czy warunek x < y jest prawdziwy:', x < y)


# Wykorzystujemy operator „większy od lub równy”, celem sprawdzenia, czy
lewa wartość jest większa lub równa od prawej

# W wyniku zostanie zwrócone „TRUE”, ponieważ lewa wartość jest większa
od prawej, nie jest równa, ale spełnia warunek

# Wynik: Czy warunek x >= y jest prawdziwy: True

print('Czy warunek x >= y jest prawdziwy:', x >= y)


# Wykorzystujemy operator „mniejszy od lub równy”, celem sprawdzenia, czy
lewa wartość jest mniejsza lub równa od prawej

# W wyniku zostanie zwrócone „FALSE”, ponieważ lewa wartość nie jest
mniejsza lub równa od prawej

# Wynik: Czy warunek x <= y jest prawdziwy: False

print('Czy warunek x <= y jest prawdziwy:', x <= y)
```

3. Operatory logiczne

W Python jak w innych językach programowania mamy do dyspozycji typowe operatory logiczne. Operatory logiczne służą do obsługi logiki w Twoich programach. Przydadzą w tworzeniu **instrukcji warunkowych if** oraz w pracy z pętlami. Najważniejsze z operatorów, z których będziesz korzystał to:

- and
- or

- not
- in
- is

Operator	Znaczenie operatora	Przykład wykorzystania
and	i (zwraca „TRUE” jeżeli oba wyrażenia są prawdziwe)	x < 3 and x < 5
or	lub (zwraca „TRUE” jeżeli jedno wyrażanie jest prawdziwe)	x < 5 or x < 3
not	negacja (zwraca „FALSE” jeżeli oba wyrażenia są prawdziwe)	not (x < 3 and x < 5)

Przykład zastosowania:

```
# Tworzymy dwie zmienne, na których sprawdzimy działanie operatorów logicznych

x  = 2

y = 4


# Wyświetlamy obie zmienne, by było nam łatwo zanalizować wynik działania poszczególnych operatorów

print('x =', x)

print('y =', y)


# Przy użyciu operatora logicznego “and” sprawdzamy, czy oba wyrażenia są prawdziwe, czyli x jest mniejsze od 3 i x jest mniejsze od 5

# W wyniku dostaniemy „TRUE”, ponieważ 2 jest mniejsze od 3 i od 5, czyli oba wyrażenia są prawdziwe

# Wynik: Czy oba wyrażenia x < 3 and x < 5 są prawdziwe: True

print('Czy oba wyrażenia x < 3 and x < 5 są prawdziwe:', x < 3 and x <5)


# Przy użyciu operatora logicznego „or” sprawdzamy, czy jedno z wyrażeń jest prawdziwe

# W wyniku dostaniemy „TRUE”, ponieważ 4 jest mniejsze od 5 i mimo, że nie jest mniejsze od 3 to warunek jest spełniony

# Wynik: Czy jedno z wyrażeń y < 5 or y < 3 jest prawdziwe: True

print('Czy jedno z wyrażeń y < 5 or y < 3 jest prawdziwe:', y < 5 or y < 3)


# Przy użyciu operatora logicznego „not” sprawdzamy czy oba wyrażenia są prawdziwe, czyli x jest mniejsze od 3 i x jest mniejsze od 5
```

```
# W wyniku dostaniemy negację „FALSE”, mimo, że oba warunki są prawdziwe, czyli 2 jest mniejsze od 3 i od 5. Zadaniem operatora „not” jest właśnie zwrócenie przeciwnej wartości logicznej do tej jaka wyszłaby ze sprawdzenia warunków bez użycia operatora "not"

# Wynik: Czy oba wyrażenia not(x < 3 and x < 5) są prawdziwe: False

print('Czy oba wyrażenia not(x < 3 and x < 5) są prawdziwe:', not(x < 3 and x <5))
```

ZADANIA

1. Napisz program, który przyjmuje jako argumenty: g -godziny, m -minuty, s -sekundy, a następnie zwraca podany czas w sekundach.
2. Wiedząc, że pierwiastek n-tego stopnia z x równa się x do potęgi 1/n, wylicz wartość pierwiastka drugiego stopnia z liczby podanej przez użytkownika.
3. Napisz program, który będzie obliczał odległości pomiędzy dwoma punktami w układzie współrzędnych. Współrzędne pierwszego punktu określone są zmiennymi x1, y1, a drugiego x2, y2. Wynik należy wydrukować poleceniem print().