Operatory arytmetyczne

Podstawy programowania w języku Python

Instrukcje i wyrażenia

- instrukcja jest operacją, którą interpreter Pythona może przetworzyć

```
1  print("Dzień dobry!")
2  x = 25
3  print(2 + 2)
```

- wyrażenie jest kombinacją wartości liczbowych i nieliczbowych, zmiennych, operatorów i wywołań fukcji (zwraca wartość)

```
1  2 + 2
2  len("Dzień dobry!")
```

Podstawowe operatory arytmetyczne

Operator	Znaczenie	Przykład	Wynik
+	dodawanie	2 + 2	4
-	odejmowanie	4 - 2	0
*	mnożenie	4 * 2	8
/	dzielenie	4 / 2	2.0
//	dzielenie całkowite	3 // 2	1
%	modulo	5 % 3	2
**	potęgowanie	3 ** 2	9

Potęgowanie

- jeśli **oba** argumenty operatora ** są liczbami całkowitymi, to wynik działania jest również liczbą całkowitą
- jeśli **choć jeden** argument operatora ** jest liczbą zmiennoprzecinkową, to wynikiem jest liczba zmiennoprzecinkowa

```
1  print(2 ** 3) # 8
2  print(2 ** 3.) # 8.0
3  print(2. ** 3) # 8.0
```

odstępstwem od tej reguły jest operator dzielenia \

Operator dzielenia

- w wyniku operacji z użyciem operatora oznaczającego dzielenie zawsze otrzymujemy liczbę zmiennoprzecinkową
- aby uzyskać w wyniku liczbę całkowitą należy zastosować operator dzielenia całkowitego //

```
1    print(6 / 3) # 2.0
2    print(6 / 3.) # 2.0
3    print(6. / 3) # 2.0
```

Operator dzielenia całkowitego

- wynik tej operacji pozbawiony jest części dziesiętnej
- w wyniku dzielenia liczby całkowitej przez liczbę całkowitą otrzymujemy wynik w postaci całkowitej
- w pozostałych przypadkach wynik będzie liczbą rzeczywistą

```
1    print(4 // 3) # 1
2    print(4 // 3.) # 1.0
3    print(4. // 3) # 1.0
```

Operator modulo

- wynikiem operacji z wykorzystaniem operatora modulo jest reszta pozostała z dzielenia całkowitego

```
1  print(14 % 4) # 2
2
3  #ponieważ:
4  print(14 // 4) # 3
5  print(3 * 4) # 12
6  print(14 - 12) # 2
```

Operatory jedno- i dwuargumentowe

Operatory + oraz – występują jako operatory:

- dwuargumentowe (dodają lub odejmują)

```
1  print(1 + 2)
2  print(4 -3)
```

- jednoargumentowe (zmieniają znak – lub nie)

```
1  print(-2) #zmiana znaku
2  print(+2) #to samo co print(2)
```

Łączenie operatorów

- łączenie operatorów determinuje szyk, wedle którego wykonywane są obliczenia w przypadku gdy pewne operatory o tym samym priorytecie zostaną zestawione ze sobą w jednym wyrażeniu
- większość operatorów wykorzystywanych przez język Python posiada łączenia lewostronne, co w praktyce oznacza, że działania wykonywane są od lewej strony do prawej

```
1  print(9 % 6 % 2) # (9 % 6) % 1 - łączenie lewostronne
2  print(2 ** 2 ** 3) # 2 ** (2 ** 3) - łączenie prawostronne
```

operator potęgowania ** wykorzystuje łączenie prawostronne!

Priorytety operatorów

Priorytet	Operator	
1	+, −	jednoargumentowe
2	**	
3	*, /, //, %	
4	+, −	dwuargumentowe

```
1    print(7 - 3 * 2) #to samo co print(7 - (3 * 2))
```

Pytanie

W wyniku operacji dzielenia liczby całkowitej przez liczbę całkowitą otrzymany wynik będzie:

a) liczbą całkowitą
b) liczbą zmiennoprzecinkową
c) ciągiem znaków
d) wartością logiczną

Odpowiedź: b)

Pytanie

Jaka będzie wartość poniższego wyrażenia?

```
3 ** 2 - 1 % 2
```

a) 4
b) 5
c) 3
d) 8

Odpowiedź: d)
Wyrażenie jest tożsame z zapisem (3 ** 2) – (1 % 2) = 9 – 1 = 8

Pytanie

Dlaczego w poniższym wyrażeniu jako pierwsze zostanie zostanie wykonana operacja mnożenia?

```
2 * 5 % 3
```

a) ponieważ operator * ma większy priorytet niż %
b) obydwa operatory mają identyczny priorytet oraz posiada łączenie lewostronne dlatego wyrażenie zostanie wykonane od lewej do prawej
c) każde wyrażenie bez nawiasów zawsze ewaluuje od lewej do prawej

Odpowiedź: b)