Operatory relacyjne, Instrukcja IF, operatory logiczne

1. Operatory relacyjne

- Wartości boolowskie są dwoma stałymi obiektami: **True (1)** i **False (0)**, używanymi do reprezentowania wartości Prawdy i Fałszu
- **None** jest używany do przedstawienia braku wartości.

Operatory porównania (relacyjne) są używane do porównywania wartości.

Poniższa tabela przedstawia działanie operatorów porównania, przy założeniu, że

x=0 , y = 1, oraz **z = 0**:

Operator	Opis	Przykład
==	zwraca True jeśli wartości operandów są równe, oraz False w przeciwnym wypadku	`x == y # False` `x == z # True`
!=	zwraca True jeśli wartości operandów NIE są równe, oraz False w przeciwnym wypadku	`x != y # True` `x != z # False`
>	zwraca True jeżeli wartość lewego operandu jest większa niż wartość prawego, oraz False w przeciwnym wypadku	`x > y # False` `y > z # True`
<	zwraca True jeżeli wartość lewego operandu jest mniejsza niż wartość prawego, oraz False w przeciwnym wypadku	`x < y # True` `y < z # False`

≥	zwraca True jeżeli wartość lewego operandu jest większa lub równa niż wartość prawego, oraz False w przeciwnym wypadku	`x >= y # False`
		`x >= z # True`
		`y >= z # True`

≤	zwraca True jeżeli wartość lewego operandu jest mniejsza lub równa niż wartość prawego, oraz False w przeciwnym wypadku	`x <= y # True`
		`x <= z # True`
		`y <= z # False`

2. Instrukcja IF

Jeśli ma zostać **wykonany fragment kody tylko wtedy, gdy spełniony jest określony warunek**, należy użyć instrukcji warunkowej **if**:

Instrukcji **if**

```
x=15
if x > 10:              # warunek jest True
    print("x > 10")
```

```
text = "Ala ma kota a kot wabi się Thomas."
if len(text) > 10:      # len – funkcja sprawdzająca długość….
    print("Napis jest dłuższy niż 10 znaków.")
```

```
number = 10
if number % 2 == 0:
    print("Liczba jest parzysta.")
```

Instrukcji **if** z następującą po niej instrukcją **else**, np.:

```
x=5
if x > 10:              # False
    print("x > 10")
else:
    print("x jest mniejsze od 10")
```

```
text1 = "Anna"
text2 = "Joanna"
if text1 == text2:
    print("Napisy są takie same.")
else:
    print("Napisy są różne.")
```

Instrukcja **if-elif-else**. Każda instrukcja **if** jest testowana osobno. Ciało **else** jest wykonywane, jeżeli ostatni **elif** ma wartość False.

```
zm1 = int(input("Podaj swój wiek ? "))

if zm1 > 18:
      print("jesteś pełnoletni masz ponad 18 lat")
      print("do 100 lat zostało ci ",100-zm1)
elif zm1 == 18:
     print("18 latek !!!")
else:
     print("dzieciak")
```

Zagnieżdżone instrukcje warunkowe, np.:

```
zm1=int(input("Podaj swoj wiek ? "))
if zm1>=18:
      print("jestes pełnoletni”)
      print("do 100 lat zostało ci ",100-zm1)
      if zm1==18:
           print("18 latek !!!- gratulacje")
else:
     print("dzieciak...")
```

Zadanie 1

Jaki jest wynik następującego fragmentu kodu?

a)

```
x = 5
y = 10
z = 8
print(x > y)
print(y > z)
```

b)

```
x, y, z = 5, 10, 8
print(x > z)
print((y - 5) == x)
```

c)

```
x, y, z = 5, 10, 8
x, y, z = z, y, x
print(x > z)
print((y - 5) == x)
```

d)

```
x = 10
if x == 10:
    print(x == 10)
if x > 5:
    print(x > 5)
if x < 10:
    print(x < 10)
else:
    print("else")
```

e)

```
x = 1
y = 1.0
z = "1"

if x == y:
    print("jeden")

if y == int(z):
    print("dwa")
elif x == y:
    print("trzy")
else:
    print("cztery")
```

3. Operatory logiczne, bitowe

Python obsługuje następujące **operatory logiczne:**

and → jeśli oba operandy są prawdziwe, warunek jest prawdziwy
or → jeśli jeden z operandów jest prawdziwy, warunek jest prawdziwy, np.
not → zwraca False, jeśli wynik jest prawdziwy, i zwraca True, jeśli wynik jest fałszywy

```
number = -7
if number > 7 or number < -7:
    print("Liczba jest większa niż 7 lub mniejsza niż -7.")
else:
    print("Liczba mieści się w przedziale od -7 do 7.")
```

Możesz **użyć operatorów bitowych** do manipulowania pojedynczymi bitami danych. Przykładowe dane:

```
x = 15, co daje 0000 1111 binarnie,
y = 16, co daje 0001 0000 binarnie.
```

zostaną użyte do zilustrowania znaczenia operatorów binarnych w Pythonie. Przeanalizuj poniższe przykłady:

& oznacza binarne **and** np. `x & y = 0` co daje 0000 0000 bin
| oznacza binarne **or** np. `x | y = 31` co daje 0001 1111 bin
^ oznacza binarne **xor**, np. `x ^ y = 31` co daje 0001 1111 bin

>> oznacza binarne przesunięcie w prawo, np. `y>>1 -> 8, co daje 0000 1000 bin`
<< oznacza binarne przesunięcie w lewo, np. `y<<3 ->128, co daje 1000 0000 bin`

Zadanie 1

Jaki jest wynik poniższego fragmentu kodu – zanim wykonasz koda zastanów się?

a)

```
x1=0b1001
x2=0b0011
print("binarne lub ",bin(x1 | x2))
print("binarne and ",bin(x1 & x2))
print("binarne xor ",bin(x1 ^ x2))
print()

x3=0b1000
print("przesuniecie bitowe w prawo ", bin(x3 >> 2))
print("przesuniecie bitowe w lewo  ", bin(x3 << 2))
```

b)

```
x = 1
y = 0
z = ((x == y) and (x == y)) or not(x == y)
print(not(z))
```

c)

```
x = 4
y = 1
a = x & y
b = x | y
d = x ^ 5
e = x >> 2
f = x << 2
print(a, b, c, d, e, f)
```