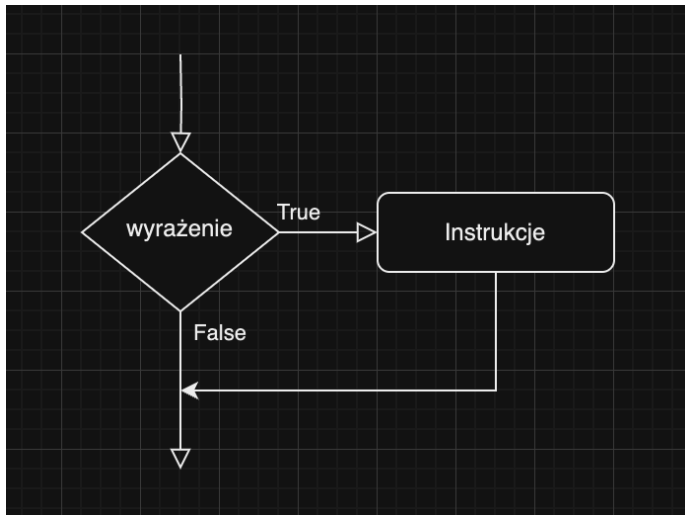


Instrukcje warunkowe

Kolejny element programowania to instrukcja warunkowa. Bardzo często używana w programowaniu. Co ta takiego?

Instrukcja warunkowa to instrukcja sprawdzająca czy „coś” jest prawdą czy fałszem. To „coś” to wyrażenie zbudowane za pomocą operatora porównania oraz operatora logicznego. Przykładem może być sprawdzenie czy podana liczba jest parzysta lub nieparzysta.

Przedstawmy w schemacie blokowym instrukcję warunkową:



Kiedy wynik wyrażenia jest **True** to wykonają się instrukcje, zaś jeśli wynik wyrażenia to **False** to opuści instrukcje i przejdzie do dalszej części programu.

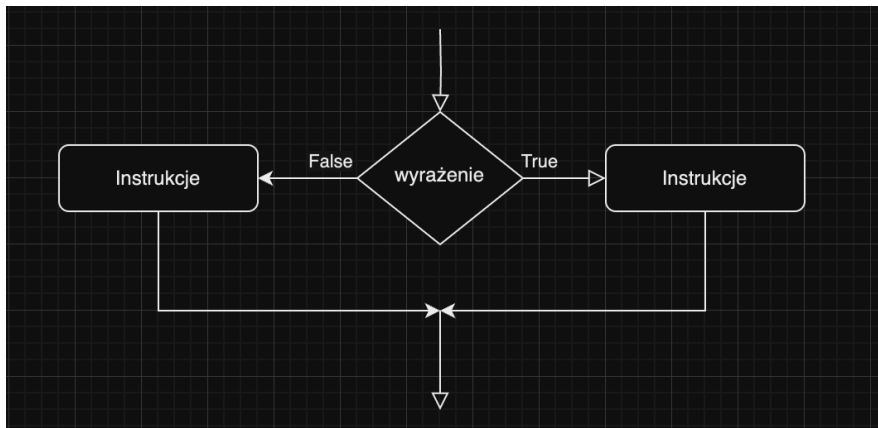
Pseudokod tego schematu wygląda następująco:

***jeżeli** liczba jest podzielna przez dwa bez reszty **to**
 jest to liczba parzysta
koniec warunku*

Kod w języku Python:

```
zmienne.py x
1  liczba = 10
2  if liczba % 2 == 0:
3      print('Liczba parzysta')
4
```

Inną opcją instrukcji warunkowej jest opcja wykonania instrukcji, kiedy wyrażenie warunkowe daje wynik fałszywy. Schemat blokowy takiej instrukcji warunkowej jest następujący:



Pseudokod tego schematu wygląda następująco:

jeżeli liczba jest podzielna przez dwa bez reszty ***to***
 jest to liczba parzysta
w przeciwnym razie
 nie jest to liczba parzysta
koniec warunku

```
1  liczba = 11
2  if liczba % 2 == 0:
3      print('Liczba parzysta')
4  else:
5      print('Liczba nie parzysta')
6
```

Dzięki takiemu rozgałęzieniu mamy dużo możliwości, opcji wykonania kodu. Nie jest możliwe wykonanie wyrażenia, jeśli nie będziemy znali operatorów porównania.

Operatory porównania

Operatory porównania możemy używać na zmiennych po swojej lewej i prawej stronie. Dzięki temu możemy sprawdzić, czy coś jest równe, różne bądź większe i podjąć decyzję co zrobić dalej. W języku Python operatorów takich mamy kilka:

> - większe od

< - mniejsze od

`>=` - większy bądź równy

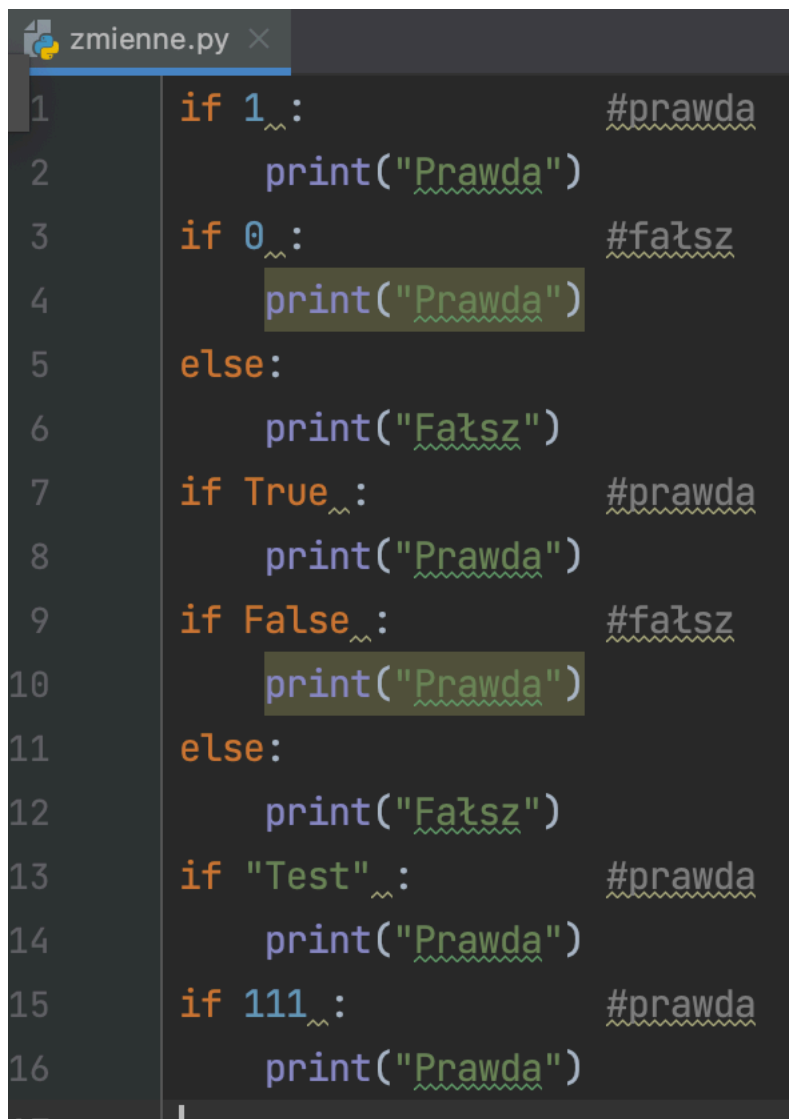
`<=` - mniejszy bądź równy

`==` - równy

`!=` - różny

Warto też wiedzieć, że wszystko, co jest różne od 0, jest prawdą.

Przykłady:



```
1  if 1:                               #prawda
2      print("Prawda")
3  if 0:                               #fałsz
4      print("Prawda")
5  else:
6      print("Fałsz")
7  if True:                           #prawda
8      print("Prawda")
9  if False:                          #fałsz
10     print("Prawda")
11 else:
12     print("Fałsz")
13 if "Test":                          #prawda
14     print("Prawda")
15 if 111:                             #prawda
16     print("Prawda")
```

Implementacja

Zacznijmy od umieszczenia w kodzie przykładowego wykorzystania instrukcji warunkowej. Zadanie będzie sprawdzało czy podana liczba jest większa od zera, zerem, czy też mniejsza od zera. Zadeklarujemy zmienną liczbę, do której przypiszemy -100. Należy pamiętać, że na zakończenie linii, w której znajduje się wyrażenie należy wstawić dwukropek `:`.

```

warunek.py x
1  liczba = -100
2  if liczba > 0:
3      print("Podana liczba jest wiejsza od zera.")
4  elif liczba < 0:
5      print("Podana liczba jest mniejsza od zera.")
6  else:
7      print("Podana liczba jest rowna zeru.")
8

```

Wcięcia

Widzimy w kodzie, że są wcięcia. Dzięki nim możemy wykonać instrukcje w bloku. Programiści Python'a są przyzwyczajeni do wykorzystywania wcięć jako zebranie instrukcji w blok, nie trzeba używać np. klamerek, nawiasów.

```

warunek.py x
1  age = 18
2  if age >= 18 :
3      print("Jesteś pełnoletni/a.")
4      print("Możesz iść na dyskotekę.")
5

```

W powyższym przykładzie jest błąd, ponieważ wcięcia są nie równe. Jakie wcięcie powinno być, pyCharm po wstawieniu tabulatora wstawia 4 spacje z automatu.

Operatory logiczne

Czasami przychodzi taki moment, że musimy ograniczyć nasz warunek z dwóch stron. Wyobraź sobie, że chcesz sprawdzić, czy ktoś jest urodzony w pomiędzy 2000 rokiem a 2010. Można to zrobić dwoma osobnymi warunkami, jednak jest to mało eleganckie rozwiązanie.

```

warunek.py x
1  year = 2006
2  if year > 2000:
3      if year < 2010:
4          print("Urodzony pomiedzy 2000 a 2010.")
5

```

Aby użyć wyrażenie w jednej linii, wykorzystamy operatory logiczne.

AND

Słowo kluczowe **and** jest operatorem logicznym i służy do łączenia instrukcji warunkowych, oba warunki muszą być prawdziwe, aby została wykonana instrukcja print, jeśli jedna z nich nie będzie prawdziwa to cały warunek zwróci fałsz.

```
warunek.py ×
1 a = 200
2 b = 33
3 c = 500
4 if a > b and c > a:
5     print("Obydwa warunki są prawdziwe.")
6
```

OR

Słowo kluczowe **or** jest operatorem logicznym i służy do łączenia instrukcji warunkowych, w tym przypadku wystarczy, aby jeden warunek był prawdziwy to całe wyrażenie warunkowa zwraca prawdę.

```
warunek.py ×
1 a = 200
2 b = 33
3 c = 500
4 if a > b or a > c:
5     print("Przynajmniej jeden z warunków ma wartość True")
6
```

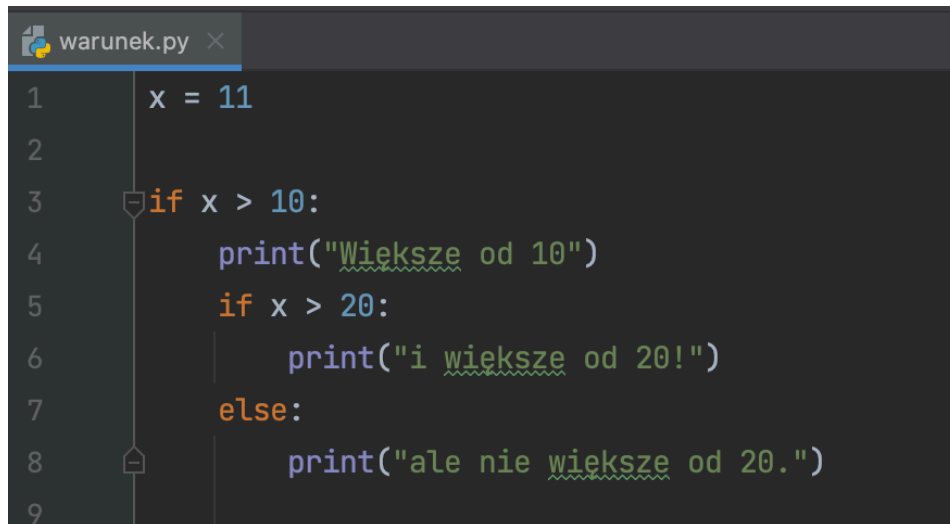
NOT

Słowo kluczowe **not** jest operatorem logicznym i służy do odwrócenia wyniku instrukcji warunkowej. Jeśli wynik wyrażenia jest prawdziwy to not zaprzecza i zwraca fałsz.

```
warunek.py ×
1 a = 33
2 b = 200
3 if not a > b:
4     print("a NIE jest większe niż b")
5
```

Zagnieżdżone Jeśli

Możesz mieć instrukcje `if` wewnątrz innej instrukcji `if`, nazywa się to instrukcjami zagnieżdżonymi. Najlepiej przedstawić to na przykładzie, sprawdź działanie tego kodu dla różnych wartości `x`:



```
warunek.py x
1 x = 11
2
3 if x > 10:
4     print("Większe od 10")
5     if x > 20:
6         print("i większe od 20!")
7     else:
8         print("ale nie większe od 20.")
9
```

Zadania do samodzielnego wykonania.