Условните конструкции в Python се използват за изпълнение на различни действия в зависимост от дадени условия. Основната условна конструкция в Python е операторът `if`. Нека разгледаме как работят основните условни конструкции.

1. Оператор `if`

Този оператор се използва за проверка на условие. Ако условието е вярно (`True`), се изпълнява блокът от код след `if`.

Пример:

```
x = 10

if x > 5:
    print("x е по-голямо от 5")
```

Тук се проверява условието: ако стойността на `x` е по-голяма от 5, програмата ще изведе съобщение.

2. Оператор `else`

Операторът `else` се използва за изпълнение на алтернативен блок код, ако условието в `if` е невярно (`False`).

Пример:

```
x = 3

if x > 5:
    print("x е по-голямо от 5")
else:
    print("x е по-малко или равно на 5")
```

Ако условието в `if` не е изпълнено, блокът код под `else` ще се изпълни.

3. Оператор `elif` (съкратено от "else if")

Този оператор позволява проверка на допълнителни условия, ако първоначалното условие в `if` е невярно.

Пример:

```
x = 7

if x > 10:
    print("x е по-голямо от 10")
elif x > 5:
    print("x е по-голямо от 5, но по-малко или равно на 10")
else:
    print("x е по-малко или равно на 5")
```

Тук програмата първо проверява дали `x` е по-голямо от 10. Ако това не е вярно, тя проверява дали `x` е по-голямо от 5. Ако нито едно от тези условия не е вярно, се изпълнява кодът в блока `else`.

4. Вложени условни оператори

Можете да използвате условни оператори един вътре в друг.

Пример:

```
x = 8
y = 3

if x > 5:
    if y > 2:
        print("x е по-голямо от 5 и y е по-голямо от 2")
```

Тук първо се проверява условието `x > 5`, след което, ако е вярно, се проверява второто условие `y > 2`.

В Python условните конструкции могат да бъдат комбинирани с **логически оператори**, за да се проверят множество условия едновременно. Най-често използваните логически оператори са:

- `and` – връща True, ако всички условия са истина.
- `or` – връща True, ако поне едно от условията е истина.
- `not` – обръща логическата стойност (връща True, ако условието е False, и обратно).

Логически оператори в условни конструкции

1. Оператор `and`

Операторът `and` се използва за проверка дали всички условия са истина. Ако поне едно от условията е False, цялото изразяване ще върне False.

Пример:

```
x = 10
y = 20

if x > 5 and y > 15:
    print("x е по-голямо от 5 и y е по-голямо от 15")
else:
    print("Едно или и двете условия не са изпълнени")
```

Тук и двете условия трябва да са истина (`x > 5` и `y > 15`), за да се изпълни първият блок код. Ако едно от тях е грешно, ще се изпълни блокът под `else`.

2. Оператор `or`

Операторът `or` се използва, когато е необходимо поне едно от условията да бъде истина, за да върне True.

Пример:

```
x = 10
y = 5

if x > 5 or y > 15:
    print("Поне едно от условията е изпълнено")
else:
    print("Нито едно от условията не е изпълнено")
```

В този случай, ако поне едно от условията е вярно (`x > 5` или `y > 15`), ще се изпълни първият блок. Ако и двете условия са грешни, ще се изпълни блокът `else`.

3. Оператор `not`

Операторът `not` обръща резултата от дадено условие. Ако условието е True, `not` го прави False, и обратно.

Пример:

```
x = 10

if not x < 5:
    print("x не е по-малко от 5")
```

Тук условието `x < 5` е False, но тъй като използваме оператора `not`, условието се обръща и става True, затова се изпълнява първият блок код.

Комбиниране на логически оператори

Можете да комбинирате няколко логически оператора за по-сложни проверки.

Пример:

```
x = 7
y = 10
z = 3

if (x > 5 and y < 15) or z == 3:
    print("Изразът е истина")
else:
    print("Изразът е лъжа")
```

Тук изразът ще бъде True, ако **или** и двете условия `x > 5` и `y < 15` са истина, **или** ако условието `z == 3` е истина. В този случай, тъй като и двете първи условия са истина, програмата ще изведе "Изразът е истина".

Приоритет на операторите

Логическите оператори имат приоритет при изпълнение:

1. Операторът `not` има най-висок приоритет.
2. Следва `and`.

3. Операторът `or` има най-нисък приоритет.

Можете да използвате скоби `()` за да промените приоритета на изпълнението или за по-добра четимост.

Пример:

```
x = 5
y = 10
z = 15

if not (x > 10 or y < 15) and z == 15:
    print("Условието е вярно")
else:
    print("Условието е грешно")
```

Тук скобите променят реда на изпълнение, като първо се изпълнява условието вътре в скобите.

Задачи

Задача 1: Проверка за четно или нечетно число

Напиши програма, която приема цяло число и проверява дали то е четно или нечетно.

Условие:
Ако числото е четно, програмата трябва да изведе "Числото е четно". Ако е нечетно, трябва да изведе "Числото е нечетно".

Подсказка:
Използвай оператора `%` (остатък при деление) за проверка дали числото се дели на 2.

```
number = int(input("Въведете число: "))

if number % 2 == 0:
    print("Числото е четно")
else:
    print("Числото е нечетно")
```

Задача 2: Проверка за положително, отрицателно или нула

Напиши програма, която приема цяло число и проверява дали то е положително, отрицателно или нула.

Условие:
Програмата трябва да изведе едно от следните съобщения в зависимост от стойността на числото:

- "Числото е положително"
- "Числото е отрицателно"
- "Числото е нула"

```
number = int(input("Въведете число: "))

if number > 0:
    print("Числото е положително")
elif number < 0:
    print("Числото е отрицателно")
else:
    print("Числото е нула")
```

Задача 3: Проверка на принадлежност към диапазон

Напиши програма, която проверява дали дадено число попада в определен диапазон от стойности.

Условие:
Програмата трябва да провери дали числото е между 10 и 50 (включително) и да изведе съответно съобщение.

```
number = int(input("Въведете число: "))

if number >= 10 and number <= 50:
    print("Числото е в диапазона от 10 до 50")
else:
    print("Числото е извън диапазона")
```

Задача 4: Проверка на години

Напиши програма, която приема годините на потребител и проверява дали той е дете, тийнейджър, възрастен или пенсионер.

Условие:

- Ако годините са между 0 и 12 (включително), програмата трябва да изведе "Дете".
- Ако годините са между 13 и 19 (включително), програмата трябва да изведе "Тийнейджър".
- Ако годините са между 20 и 64 (включително), програмата трябва да изведе "Възрастен".
- Ако годините са 65 или повече, програмата трябва да изведе "Пенсионер".

```
age = int(input("Въведете годините: "))

if age >= 0 and age <= 12:
    print("Дете")
elif age >= 13 and age <= 19:
    print("Тийнейджър")
elif age >= 20 and age <= 64:
    print("Възрастен")
elif age >= 65:
    print("Пенсионер")
else:
    print("Невалидна възраст")
```

Задача 5: Проверка на парола

Напиши програма, която проверява дали въведената парола съответства на зададена стойност.

Условие:
Програмата трябва да приеме парола от потребителя и да провери дали тя съответства на предварително зададената парола "mypassword123". Ако паролата е вярна, изведи "Достъпът е разрешен", а ако е грешна — "Грешна парола".

```
correct_password = "mypassword123"
entered_password = input("Въведете паролата: ")

if entered_password == correct_password:
    print("Достъпът е разрешен")
else:
    print("Грешна парола")
```

Задача 6: Проверка за високосна година

Напиши програма, която проверява дали дадена година е високосна.

Условие:

- Една година е високосна, ако се дели на 4, но не се дели на 100, освен ако не се дели на 400.

Пример:
2020 е високосна година, но 1900 не е, защото се дели на 100, но не на 400. 2000 обаче е високосна.

```
year = int(input("Въведете година: "))

if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
    print(f"{year} е високосна година")
else:
    print(f"{year} не е високосна година")
```
