Операторы ветвления. Множественное ветвление

Операции сравнения

Операции сравнения - сравнивают два значения. И по результату возвращают True или False.

Python поддерживает следующие операции сравнения:

== Возвращает True, если оба операнда **равны**. Иначе возвращает False.

!= Возвращает True, если оба операнда **НЕ равны**. Иначе возвращает False.

> (больше чем) Возвращает True, если **первый операнд больше второго**.

< (меньше чем) Возвращает True, если **первый операнд меньше второго**.

>= (больше или равно) Возвращает True, если первый операнд **больше или равен второму**.

<= (меньше или равно) Возвращает True, если первый операнд **меньше или равен второму**.

Операции сравнения

== Возвращает True, если оба операнда **равны**. Иначе возвращает False.

Наша цель найти равные элементы.

Пример:

```
a = 11
b = 5
result = a== b  # сохраняем результат операции в переменную
print(result)  # False - 11 не равно 5

bool1 = True
bool2 = True
print(bool1 == bool2)  # True - bool1  равно bool2

b1 = «Строка»
b2 = «Строка2»
print(b1 == b2)  # False - b1  не равно b2
```

Операции сравнения

!= Возвращает True, если оба операнда **НЕ равны**. Иначе возвращает False.

Наша цель найти НЕ равные элементы.

Пример:

```
a = 11
b = 5
result = a != b  # сохраняем результат операции в переменную
print(result)  # True – 11 действительно не равно 5

c = 10
d = 10
print(c!= d)  # False – потому что 10 равно 10
```

Операции сравнения

> (больше чем) Возвращает True, если **первый операнд больше второго**.

Наша цель проверить больше ли первый элемент чем второй.

Пример:

```
a = 11
b = 5
result = a > b
print(result)  # True – 11 действительно больше 5

print(b>a)  # False – потому что 5 меньше 11
```

Операции сравнения

< (меньше чем) Возвращает True, если **первый операнд меньше второго**.

Наша цель проверить меньше ли первый элемент чем второй.

Пример:

```
a = 11
b = 5
result = a < b
print(result)  # False – 11 не больше 5

print(b<a)  # True – потому что 5 меньше 11
```

Операции сравнения

>= (больше или равно) Возвращает True, если **первый операнд больше или равен второму**.

Наша цель проверить больше или равен первый элемент второму.

Пример:

```
a = 11
b = 5
print(a >= b)  # True – 11 больше 5
print(b>=a)  # False – потому что 5 не больше чем 11 и не равно ему
b = 11
print(b >= a)  # True – 11 не больше чем 11, но равно ему, по этому  True
```

Операции сравнения

<= (меньше или равно) Возвращает True, если **первый операнд меньше или равен второму**.

Наша цель проверить меньше или равен первый элемент второму.

Пример:

```
a = 5
b = 11
print(a <= b)  # True – 5 меньше 11
print(b <= a)  # False – потому что 11 не меньше чем 5 и не равно ему
b = 5
print(b <= a)  # True – 5 не меньшее чем 5, но равно ему, по этому  True
```

Логические операции

Логические операции используются для создания сложных(составных) логических конструкций. Например, когда надо объединить два условия.

Python поддерживает следующие логические операции:

and (логическое умножение)
Возвращает True, если оба выражения равны True.

or (логическое сложение)
Возвращает True, если хотя бы одно из выражений равно True.

not (логическое отрицание)
Возвращает True, если выражение равно False.

Логические операции

and (логическое умножение) Возвращает True, если оба выражения равны True.

Наша цель удостоверится, что все условия объединенные and истины или выполняются.

```
a = 5
b = 8
#Случай 1 оба выражения истина
result = a > 4 and b== 8
print(result)  # True- потому что и 5> 4 и 8 равно 8
#Случай 2 не выполняется первое выражение
result = a >6  and b== 8
print(result)  # False не смотря на то, что 8 равно 8, потому что 5 меньше 6
#Случай 3 не выполняется второе выражение
result = a >4  and b== 7
print(result)  # False не смотря на то, что 5 больше 4, потому что 8 не равно 7
#Случай 4 ничего  не выполняется
result = a >6  and b== 7
print(result)  # False – потому что 5 меньше 6 и 8 не равно 7
```

Логические операции

or (логическое сложение) Возвращает True, если хотя бы одно из выражений True.

Наша цель удостоверится, что хотя бы одно условие истинное

```
a = 5
b = 8
#Случай 1 оба выражения истина
result = a > 4 or b== 8
print(result)  # True- потому что и 5> 4 и 8 равно 8
#Случай 2 не выполняется первое выражение
result = a >6  or b== 8
print(result)  # True 8 равно 8 – этого достаточно, первое выражение ложно
#Случай 3 не выполняется второе выражение
result = a >4  or b== 7
print(result)  # True  5 больше 4– этого достаточно, второе выражение ложно
#Случай 4 ничего  не выполняется
result = a >6  or b== 7
print(result)  # False – потому что 5 меньше 6 и 8 не равно 7
```

Логические операции

not (логическое отрицание) Возвращает True, если выражение равно False.

```
a = 9
b = 8
#Случай 1
result = a > b # это выражение True
print(not result)  # False – было True – стало False
#Случай 2
print(not a<2)  # True – выражение 9<2 – ложно
```

Условная конструкция if

Условная конструкция (оператор ветвления, условный оператор) — оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.

```
if логическое_выражение:
    инструкции
[elif логическое выражение:
    инструкции]
[else:
    инструкции]
```

Условная конструкция if (Пример 1)

```
кол-во_учащихся=24
if (кол-во_учащихся<25):
    print(“На паре не все учащиеся.”) #Выполниться только если условие в if истинное.
print(“Паре быть!”) #Выполниться в любом случае.
```

Результат:

На паре не все учащиеся.

Паре быть!

```
кол-во_учащихся=25
if (кол-во_учащихся<25):
    print(“На паре не все учащиеся.”)
print(“Паре быть!”)
```

Результат:

Паре быть!

Условная конструкция if (Пример 2)

(влияние отступов)

```
кол-во_учащихся=25
if (кол-во_учащихся<25):
    print(“На паре не все учащиеся.”)
print(“Паре быть!”)
```

Результат:

Паре быть!

```
кол-во_учащихся=25
if (кол-во_учащихся<25):
    print(“На паре не все учащиеся.”)
    print(“Паре быть!”)
```

Результат:

Условная конструкция if (Пример 3)

(Конструкция с else)

```
кол-во_учащихся=24
if (кол-во_учащихся<25):
    print(“На паре не все учащиеся.”) #Выполниться если условие в if истинное.
else:
    print(“Паре быть!”) #Выполниться если условие в if ложное.
```

Результат:

На паре не все учащиеся.

```
кол-во_учащихся=25
if (кол-во_учащихся<25):
    print(“На паре не все учащиеся.”)
else:
    print(“Паре быть!”)
```

Результат:

Паре быть!

Условная конструкция if (Пример 4)

(Конструкция с else)

```
кол-во_учащихся=??
if (кол-во_учащихся<25):
    print(“На паре не все учащиеся.”) #Выполниться если условие в if истинное
    print(“Никто не хочет ходить на пары.”)
    print(“Преподавателю грустно”)
else:
    print(“На паре все учащиеся”) #Выполниться если условие в if ложное.
    print(“Преподаватель доволен”)
print(“Пара будет в любом случае.”) #Выполниться в любом случае.
```

Условная конструкция if (Пример 5)

(Вложенные конструкции)

```
кол-во_учащихся=??
if (кол-во_учащихся<25):
    print(“На паре не все учащиеся.”)
    print(“Пара будет!”)
    if (кол-во_учащихся<12):
        print(“Пару надо отменить(”)
    else:
        print(“Преподаватель не доволен”)
```

Условная конструкция if (Пример 6)

(Вложенные конструкции)

```
кол-во_учащихся=??
if (кол-во_учащихся<25):
    print(“На паре не все учащиеся.”)
    print(“Пара будет!”)
 elif (кол-во_учащихся<12):
     print(“Пару надо отменить(”)
  else:
      print(“Преподаватель не доволен”)
```

Рассмотрим пример.

Реализуйте подсчет выплаты от доходов, учитывая, что если доходов больше 100р, то выплата должна состоять 20% от суммы, если доходов больше 60р, то выплата должна состоять 25%, в противном случае - 30%.

Решение примера:

```
col=int(input("Введите денюжку "))
if(col>=100):
    dd=0.2*col
elif(col>=60):
    dd=0.25*col
else: dd=0.3*col
print('ваш заработок',dd)
```