Теоретический материал к занятию Условия

Программы должны уметь выполнять разные действия в зависимости от введенных данных. Для принятия решения программа проверяет, истинно или ложно определенное условие. Проверка условия и принятие решения по результатам этой проверки называется **ветвлением.**

В Python проверка условия осуществляется при помощи условного оператора if.
В условиях строка заканчивается символом двоеточия (:). Этот символ указывает на начало блока текста кода, который относится к условному оператору. В блок входят ВСЕ команды, расположенные с отступом. Если условие истинно, то выполняется весь блок команд.

```
x = 5
if x == 5:
   print('x равно')
```

Предыдущий код выводит текст в случае если условие истинно. Но если условие ложно, то программа ничего не выводит. Чтобы выполнять что-либо, если условие оказалось ложным, мы используем ключевое слово **else:** *(иначе)*

```
x = 5
if x == 5:
   print('x равно')
else:
   print('x не равно')
```

В языке Python: отступы являются обязательными и по сути заменяют специальные символы и ключевые слова. Создатели языка Python посчитали, что такие отступы делают код более чистым и его будет намного проще читать. **Отступ** - небольшое смещение строки вправо. **Отступ** сообщает где начинается и заканчивается блок кода. Согласно стандарту PEP 8, для отступа используется расстояние ровно в 4 пробела или 1 TAB.

В Python существует 6 основных типов проверки, их называют **операторами сравнения**.

```
x == 5   # равенство
x > 5    # больше
x < 5    # меньше
x >= 5   # больше или равно
x <= 5   # меньше или равно
x != 5   # не равно
```

Условия могут быть и вложенные друг в друга, составляя цепочку проверок

```
x = 5
```

```
if x > 2:
    if x < 10:
        print('x больше 2, но меньше 10)
```

В Python есть три логические операции, которые помогают создавать сложные логические условия. Логическое условие называется сложным если оно состоит из 2 и более условий, соединенных одной из логических операций:

1. *and: логическое умножение;*
2. *or: логическое сложение;*
3. *not: логическое отрицание.*

```
age = 12
user_class = 7
if age >= 12 and user_class == 7:
   print('Условие верно')
```

Логическое умножение, то есть операция and, истинно только когда оба условия вокруг and - истинны. Если хотя бы одно из значений ложно, то и результат будет ложным. Другими словами для того, чтобы сложное логическое условие составленное из логических and было истинным, нужно, чтобы все условия были истинным.

```
age = 12
user_class = 7
if age >= 12 or user_class == 7:
   print('Условие верно')
```

Данная операция также позволяет создавать сложные условия, но у нее есть отличие от and. Для истинности сложного условия необходимо, чтобы ХОТЯ БЫ одно из условий в составе сложного условия было истинным. Именно ХОТЯ БЫ одно, а не все, как это было у and.

Логическая операция or как и and позволяет объединять сколько угодно условий.

Связка if..else, подходит только когда у нас не более двух значений , но что если у нас больше двух вариантов. Конечно же мы можем написать программу и с помощью if, но стоит запомнить, что это будут отдельные условия и может быть вариант, когда несколько из них окажутся верными.

```
x = 5
if x >= 5:
   print('YES')
if x == 5:
   print('YES')
if x != 10:
   print('YES')
```

```
else:
   print('NO')
```

Что же делать, когда у нас более 2 вариантов и как написать условие? Для этого существует оператор elif *(если иначе)*. Он позволяет составлять длинные условия с неограниченным количеством вариантов.

```
age = int(input('Введите свой возраст: '))
if age >= 14:
  print('Вам больше 14 или 14 лет')
elif age >= 18:
  print('Вам больше или 18 лет')
elif age >= 21:
  print('Вам больше или 21 год')
elif age > 45:
  print('Вам больше 45 лет')
```

Если требуется проверить значение по двум или более параметрам у нас есть несколько вариантов написание подобного кода.

1. Через вложенные условия.
2. Используя операторы and или or
3. Используя проверку 'по диапазону'

Разберем все варианты.

```
age = 12
if age > 11:
   if age < 15:
      print('Да')
```

```
age = 12
if age > 11 and age < 15:
   print('Да')
```

```
age = 12
if 11 < age < 15:
   print('Да')
```

Первый и второй варианты нам понятны, но как работает третий вариант. Читается он как если age больше 11 и меньше 15. В данном случае переменная всегда будет стоять между проверочными значениями.