Методические указания

Урок 3.1 Условный оператор. Логические операции and, or, not

Задачи урока:

- Познакомить с условным оператором if: else:
- Познакомить с логическими операциями and, or, not

0. Подготовка к уроку

До урока учителю необходимо:

1) Просмотреть в классе как ученики справились с домашним заданием
2) Прочитать методичку

1. Знакомство с условным оператором

Учитель: На предыдущих уроках мы познакомились с базовыми строительными блоками программ. Теперь мы умеем писать программы, обеспечивающие ввод, обработку и вывод данных. Более того, мы умеем работать с строками и числами, подобно тому как мы делаем это в математике. Пришло время познакомиться со способами управления ходом выполнения программы.

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.

Учитель: Обратите внимание, что 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')
```

Предыдущая программа 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')
```

Если выполняется условие в строке if, то выполняется первый блок команд, иначе если не выполняется, то выполняется второй блок команд.

Учитель: Во многих языках программирования (Pascal, C++, Java, C#) отступы - это дело вкуса программиста, он может их ставить, а может не ставить, главное ставить специальные символы, такие как скажем фигурные скобки {} (C++, C#, Java) или ключевые слова begin end (Pascal). Другое дело 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 таб.

Учитель: Вы возможно заметили, что в предыдущей программе для проверки мы использовали символ ДВОЙНОЕ РАВНО. Это была не ошибка. Действительно, когда мы хотим, проверить на равенство мы пользуемся оператором сравнения, который обозначается именно двойным равно ==. Оператор == называется оператором сравнения и проверяет на равенство два значения.

Не стоит путать оператор присваивания (одно равно =) и оператор сравнения (два равно ==). Эти операторы работают по-разному и предназначены для разных вещей.
Оператор присваивания (один символ =) придает переменным значение, в то время как оператор сравнения (==) проверяет на равенство. Оператор == используется всегда для сравнения элементов.

Ученик может спросить, а зачем придумали двойное равенство: нужно сказать, что Python как-то должен отличать процедуру присвоения и проверки поэтому и стали использовать удвоенный знак (= и ==).

Следует сказать, что путаница с этими операторами одна из самых частых у начинающих программистов и если ваша программа не работает, то первым делом нужно посмотреть именно на это.

Учитель: Как вы понимаете проверка в условном операторе может быть не только на равенство. 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
```

```
num1 = 3
num2 = 7
if num1 < num2:
    print('Меньше')
else:
    print('Больше')
```

Далее спрашиваем учеников какой будет правильный ответ в данном коде, если num1 = 3, num2 = 7? *Правильный ответ: Меньше*

Учитель: одним из преимуществ языка Python является много встроенных и реализованных фич. Фича на сленге программистов означает крутой функционал, который позволяет упростить написание кода. Так вот, в языке Python есть так называемые цепочки сравнений, которые позволяют объединять несколько условий как бы в одно.

```
x = 5
if x > 2:
     if x < 10:
```

```
        print('x больше 2, но меньше 10)
```

.

2. Логические операции and, or, not

Учитель Как быть в ситуации, когда у нас есть несколько условий? Если условий всего два, то могут помочь цепочки сравнения, и то не всегда.

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.

Учитель: Давайте начнем с самой часто используемой логической операции: and. Предположим, мы хотим написать программу для учеников от двенадцати лет, которые учатся по крайней мере в 7 классе. Таким образом у нас есть два условия и нужно проверить, что оба условия одновременно выполняются. В таком случае мы используем логическую операцию and и объединяем оба наших условия в одно сложное.
Логическая операция and позволяет объединять сколько угодно условий.

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

Учитель: 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')
```

Учитель: 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.

Учитель: Логическое сложение, то есть операция or ложна только когда оба выражения a и b - ложны. Если хотя бы одно из значений истинно, то и результат будет истинным. Другими словами для того, чтобы сложное логическое условие составленное из логических or было истинным, нужно, чтобы хотя бы одно из условий было истинным.

```
age = 12
if not age == 12:
   print('Возраст не равен 12')
```

Существует еще одна логическая операция, которая называется логическим отрицанием и обозначается not. Операция not позволяет указать противоположный результат логического выражения.

Учитель: Подобно арифметическим операциям, логические операции тоже имеют приоритет, порядок применения, как и в математике.
Для того, чтобы изменить приоритет, мы используем скобки, так же как и в арифметических операциях.

3. Решение задач

Задача 1
Написать программу которая запрашивает на ввод температуру тела человека и выводит: здоров, низкая температура, высокая температура

Решение

```
temp = float(input('Введите температуру '))
if temp <= 36.6:
    print('Здоров')
if temp <= 37.5:
    print('Низкая температура')
if temp > 38:
    print('Высокая температура')
```

Дополнительно

Если на уроке остается время, то ученикам можно предложить начать прорешивать домашнее задание.

Домашняя работа

1.