**Информация об уроке**

**Модуль: 1**

**Номер занятия: 9**

**Тип занятия: Практика**

**Название занятия/ Тема занятия:** Условные выражения и конструкции, отладка кода.

**Цель занятия:** изучить базовые настройки Python.

**Образовательные результаты:** Изучить переменные, их типы данных и базовые операторы.

**Глоссарий:**



**Конспект занятия**

1. **Приветствие**

Приветствуем вас на новом курсе по Python!

1. **Тема урока и целеполагание**

Сегодня мы пройдём:

1. **Условные выражения**
2. **Отладка кода**
3. **Актуализация**

Для начала вспомним материал прошлого урока:

Ряд операций представляют условные выражения. Все эти операции принимают два операнда и возвращают логическое значение.

Простейшие условные выражения представляют операции сравнения, которые сравнивают два значения. Python поддерживает следующие операции сравнения:

| **Оператор** | **Описание** | **Пример** |
| --- | --- | --- |
| == | Возвращает True, если оба операнда равны | a == b |
| != | Возвращает True, если оба операнда не равны | a != b |
| > | Возвращает True, если первый операнд больше второго | a > b |
| < | Возвращает True, если первый операнд меньше второго | a < b |
| >= | Возвращает True, если первый операнд больше или равен второму | a >= b |
| <= | Возвращает True, если первый операнд меньше или равен второму | a <= b |

Для создания составных условных выражений применяются логические операции. В Python имеются следующие логические операторы:

**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

**Оператор in**

Оператор in возвращает True если в некотором наборе значений есть определенное значение. Он имеет следующую форму.

Например, строка представляет набор символов. И с помощью оператора in мы можем проверить, есть ли в ней какая-нибудь подстрока.

Если нам надо наоборот проверить, нет ли в наборе значений какого-либо значения, то мы можем использовать модификацию оператора - **not in**. Она возвращает True, если в наборе значений НЕТ определенного значения:

Условные конструкции используют условные выражения и в зависимости от их значения направляют выполнение программы по одному из путей. Одна из таких конструкций - это конструкция **if**, если это логическое выражение возвращает True, то выполняется последующий блок инструкций, каждая из которых должна начинаться с новой строки и должна иметь отступы от начала выражения if (отступ необходимо делать в 4 пробела).

Если вдруг нам надо определить альтернативное решение на тот случай, если выражение в if возвратит False, то мы можем использовать блок **else**.

Если необходимо ввести несколько альтернативных условий, то можно использовать дополнительные блоки **elif**, после которого идет блок инструкций.

Сначала Python проверяет выражение if. Если оно равно True, то выполняются инструкции из блока if. Если это условие возвращает False, то Python проверяет выражение из elif.

Если выражение после elif равно True, то выполняются инструкции из блока elif. Но если оно равно False то выполняются инструкции из блока else

При необходимости можно определить несколько блоков elif для разных условий. Например:

Конструкция **if** в свою очередь сама может иметь вложенные конструкции if (условия в условии).

Отладкой кода называют процесс последовательного приведения кода в рабочее состояние после нахождения ошибки. Существуют различные методы отладки:

* **Debug** - выполнение программы в режиме действие за действием, чтобы напрямую вычислить ошибку (доступно только в профессиональных средах программирования)
* **Проверки выводом** - добавление функций print() в разные части программы, чтобы видеть состояния переменных и программы в тот или иной момент времени выполнения.

1. **Основное содержание**

**ЗАДАЧИ:**

**1)** Пожилой учитель Николай Семёнович выставляет в журнал оценки. Он плохо видит, поэтому иногда случайно нажимает на неподходящие для оценок цифры.

Напиши программу-фильтр, которая принимает число и выводит “Оценка (число)”, только если оно меньше или равно 5, а иначе выводит предупреждение: “Слишком большое число”.

Пример:

Ввод: 4 Вывод: Оценка: 4

Ввод: 7 Вывод: Слишком большое число

**Решение**

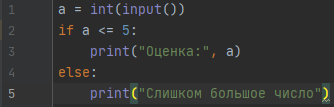
Для решения этой задачи построим мини модель:

Все числа <= 5:

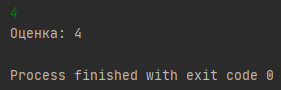
выводим   
иначе:

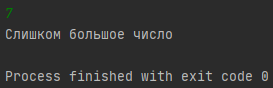
выводим сообщение

Переведем эту мини модель в код:



Консольные выводы:





**2)** Боря покупал конфеты ко дню рождения. Он набрал сначала А конфет, потом вспомнил, что человек на празднике будет d и нужно, чтобы всем досталось поровну.

Даны A,d. Выведи ближайшее к А число, кратное d. Если ближайших несколько, выведи меньшее.

Пример:

Ввод: 13 Вывод: 15

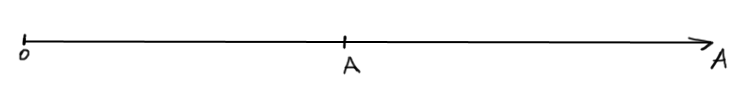
5

Ввод: 15 Вывод: 10

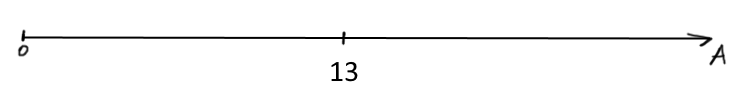
10

**Решение**

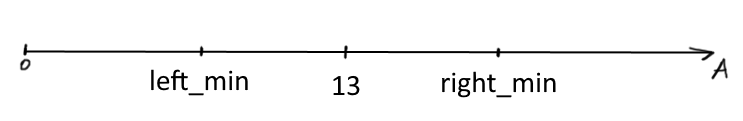
Для начала надо найти кратные **d** числа, рядом с **А**. Представим числовую прямую:



Для лучшего понимания разберем первый пример ввода, затем вместо чисел подставим переменные:



Нам необходимо найти левое минимальное число, кратное **5** и правое минимальное число, кратное **5**:



Чтобы найти **left\_min** достаточно целочисленно поделить **13** на **5**:

**13 // 5 = 2**

Затем следует умножить это число на **5**, тем самым мы отсекаем лишний промежуток между **10** и **13**:

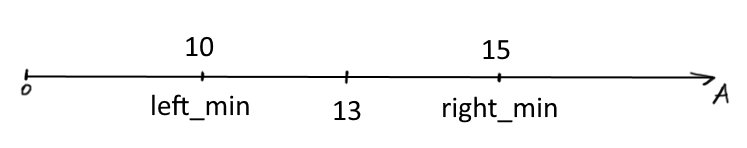
**left\_min = 13 // 5 \* 5 #10**

Чтобы найти **right\_min** следует осознать, что эта переменная на **5** больше, чем **left\_min** так как это следующее число, кратное **5**:

**right\_min = 13 // 5 \* 5 + 5**

либо

**right\_min = (13 // 5 + 1) \* 5**



На заключительном этапе необходимо сравнить какая разница будет меньше:

* между 13 и 10
* между 13 и 15

По условию надо вывести число с наименьшей разницей и если разница одинакова, вывести меньшее. Составим условие словесно:

**Если 13 - 10 <= 15 - 13:**

**вывести 10**

**Иначе:**

**вывести 15**

Заменим все числа на переменные:

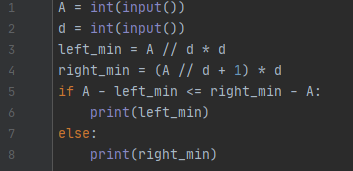
**13 = A**

**5 = d**

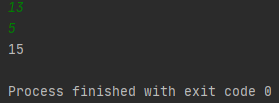
**13 // 5 \* d = left\_min = 10**

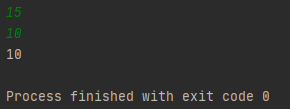
**(13 // 5 + 1) \* 5 = right\_min = 15**

Получим:



Консольные выводы:





**3)** Петя и Вася поспорили, делится ли n на k или не делится.

Напиши программу, которая решает их спор.

Если n не делится на k, программа должна вывести строку из восклицательных знаков, длина которой равна остатку от деления. А если делится без остатка, нужно вывести пустую строку

Пример:

Ввод: 7 Вывод: !!

5

Ввод: 8 Вывод:

4

**Решение**

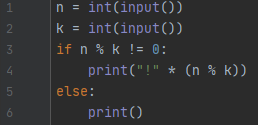
По условию нас спрашивают делится ли одно число на другое. Для того, чтобы понять как подобрать операцию, вспомним что вообще значит одно число делится на другое.

Это означает, что после деления нет остатка или же перефразируем так:  
После деления одного числа на другое остаток равен 0.

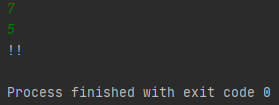
Поскольку в определении деления фигурирует остаток, будем использовать операцию остаток от деления %.

Второй немаловажный момент - необходимо вывести столько восклицательных знаков, чему равен остаток от деления. Вспомним также, что строки можно умножать на числа.

Получим такую программу:



Консольный вывод:



**4)** Женя не любит деление с остатком. Ей кажется несправедливым, что получается так мало, особенно когда дело касается конфет, поэтому она всегда частное округляет до большего целого значения.

Даны два числа - а и b. Выведи результат деления a на b, округленный наверх.

Пример:

Ввод: 5 Вывод: 2

3

**Решение**

Разберём задачу на примере, данном в условии.

Поделим 5 на 3:

5 / 3 = 1,666…

5 % 3 = 2

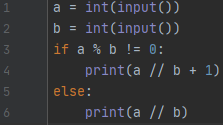
5 // 3 = 1

Брать обычное деление для решения данной задачи нецелесообразно, поскольку нам придется избавляться от дробной части путем округления, где мы не знаем в большую или меньшую сторону Python решит округлить.

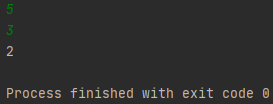
При остатке не равном нулю, мы будем к целочисленному делению прибавлять 1, тем самым отбросив остаток и округлив по условию

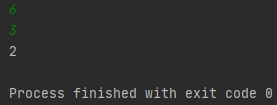
При остатке равном нулю, мы будем выводить только результат целочисленного деления.

Получим код:



Консольные выводы:





**5)** Рассеянный Витя никак не может запомнить имена троих родственников, которые приехали к ним домой на новый год.

Даны подряд возрасты и имена трех родственников (возраст, имя, возраст, имя, возраст, имя). Выведи имя среднего по возрасту родственника.

Пример:

Ввод: 3 Вывод: Лёня

Петя

29

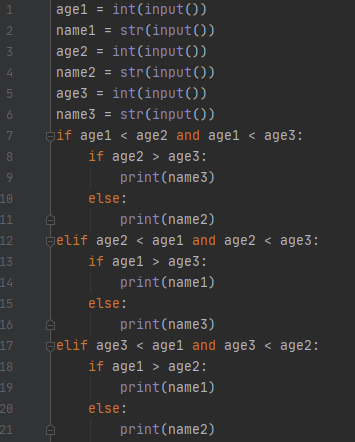
Надя

15

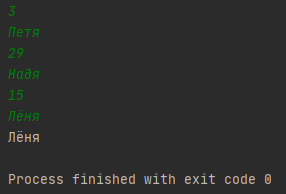
Лёня

**Решение**

Нам даны 3 возраста с привязкой к 3 именам. Для начала определяем средний возраст. Для этого найдем самый маленький возраст и самый большой. Воспользуемся для этого вложенными условными конструкциями:



Консольный вывод:



**6)** Саша и Сеня написали контрольную на 5, Катя и Коля на 4, а Наташа и Никита не пришли. По имени ученика или ученицы выведи оценку, если известно, что за прогул ставят 2.

Пример:

Ввод: Никита Вывод: 2

**Решение**

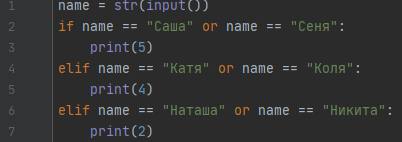
Для решения этой задачи создадим много условий, по которым будем выводить правильные оценки. Вводные данные такие:

Саша или Сеня - 5

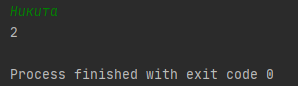
Катя или Коля - 4

Наташа или Никита - 2

Получим код:



Консольный вывод:



**7)** Даны три числа. Выведи их через пробел от меньшего к большему.

Пример:

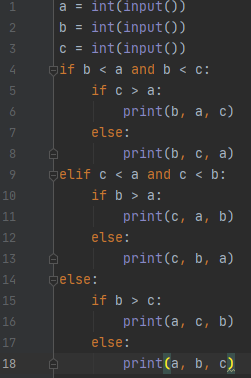
Ввод: 5 Вывод: 2 5 7

2

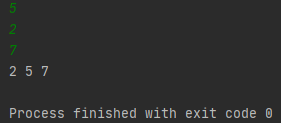
7

**Решение**

Для решения этой задачи необходимо найти минимальное число и максимальное, оставшееся обязательно будет средним. за минимальное и максимальное число изначально берём первое. Перебрав все возможные варианты сравнения, получим:



Консольный вывод:



1. **Подведение итогов/Рефлексия**

На этом занятии мы узнали что такое программа, из чего она состоит. Какой синтаксис в языке программирования Python, на что обращать внимание при написании кода. Что такое переменные и типы данных, какие операции существуют в языке и как они работают.