Условия

Условия это, наверное, душа программирования. Мы очень часто хотим, чтобы наша программа работала по разному в зависимости от происходящего. Алгоритмы, использующие условия, называют разветвляющимися, и для их понимания очень удобно рисовать блок-схемы (нет, они не остались в школе, если в коде много условий и разных действий по-прежнему очень помогает нарисовать на бумаге дерево).

Условия (if) позволяют выполнять код только в случае истинности какого-то логического выражения.

Проще говоря, "если верно, что..., то сделать ...".

Самый простой пример использования if - это вывод какой-то фразы по условию.

In [3]:
```
x = 1
if x == 1:  # Выражение равно True, это условие истинное
    print('That is true!')  # Фраза выводится
```

```
That is true!
```

In [4]:
```
if x != 1:  # Выражение равно False, это условие ложное
    print('That is true!')  # Фраза не выводится
```

Обратите внимание, что код, который находится внутри условия, выделяется отступом в 4 пробела или табуляцией (работает не во всех IDE, но в Jupyter все будет хорошо).

Иначе программа не поймет, что он относится к условию.

In [5]:
```
if x == 1:
print('That is true!')
```

```
  File "<ipython-input-5-b1fb9bc19953>", line 2
    print('That is true!')
        ^
IndentationError: expected an indented block
```

А что делать, если в том случае, когда условие не истинное, мы тоже хотим совершать какое-то действие? Для этого у нас есть ключевое слово **else ("то")**.

In [6]:
```
if x != 1:
    print('That is true!')
else:
    print('That is false!')
```

```
That is false!
```

Мы разобрались, как поступать, если у нас два варианта действий, но их может быть и больше.

Для примера давайте решим простую задачу - найти минимум из двух введенных чисел. Пока ничего нового:

In [7]:
```
a = input('Введите первое число: ')
b = input('Введите второе число: ')
if a < b:
    min = a
else:
    min = b
print('Минимум равен', min)
```

```
Минимум равен 3
```

А теперь усложним задание, добавив третий вариант развития событий - если числа равны, будем печатать '*Равные числа*'.

Можно решить эту задачу с помощью вложенных условий:

In [9]:
```
a = input('Введите первое число: ')
b = input('Введите второе число: ')
if a < b:
    print(a)
else:
    if a > b: # обратите внимание, здесь одно условие находится внутри другого, и ко
        print(b)
    else:
        print('Равные числа:', a)
```

```
Равные числа: 3
```

Неплохо, но можно упростить это решение с помощью конструкции **else if (или elif)**, которая позволяет в случае ложности условия сразу же написать еще одну проверку.

Вот как будет выглядеть решение нашей задачи с помощью elif:

In [23]:
```
a = input('Введите первое число: ')
b = input('Введите второе число: ')
if a < b:
    print(a)
elif a > b:
    print(b)
else:
    print('Равные числа:', a)
```

```
Введите первое число: 3
Введите второе число: 3
Равные числа: 3
```

Задачи для тренировки

Распродажа

В магазине проходит акция:

- На все товары дешевле 1000 рублей скидка 15%
- На все товары дороже 1000, но дешевле 5000 рублей скидка 20%
- На все товары дороже 5000 рублей скидка 25%

Ввод

Целое неотрицательное число - цена товара в рублях

Вывод

Целое неотрицательное число - скидка на товар в рублях

In [12]:
```
# место для решения
price = int(input())

if price > 5000:
    print(price * 0.25)
elif price > 1000:
    print(price * 0.20)
else:
    print(price * 0.15)
```

```
149.85
```

Цикл while

Довольно часто задачи требуют от нас несколько раз выполнить однотипный код.

Если писать несколько раз одни и те же строки, это загромождает программу. Иногда несколько раз превращается в много (100 или 10000). А иногда это число вообще зависит от параметров ввода.

Справиться с этим помогают **циклы**. На этом семинаре мы поработаем с циклом **while (пока)**

Прицип использования цикла while: записываем логическое выражение и некоторый код. Код будет выполняться до тех пор, пока логическое выражение верно.

Например, давайте напечатаем все целые числа от 1 до 10.

In [10]:
```
i = 1
while i <= 10:
    print(i)
    i += 1
```

```
1
2
3
4
5
6
7
8
9
10
```

Здесь мы использовали запись i += 1. Она эквивалентна i = i + 1.
Аналогично можно записывать и другие арифметические операции: например, -=

Обратите внимание, что код внутри цикла (тот, который мы хотим повторно выполнять), выделяется отступом.

Операторы break и continue.

Циклами можно управлять с помощью операторов **break**, **continue**.

Break внутри цикла позволяет прервать его выполнение и сразу же перейти к коду, который идет после цикла (либо завершить программу).

В этом случае мы можем написать сразу после цикла секцию **else** (синтаксис при этом такой же, как и в условиях).
Код, написанный после **else**, будет выполняться, если цикл завершился "естественным путем" (т.е. не был прерван с помощью break).

Задача

Рассмотрим пример задачи, которую можно решить с использованием break.

Пусть студент сдал 5 предметов во время сессии и мы хотим узнать, есть ли у него пересдачи

Формат ввода
До пяти оценок от 1 до 10

Формат вывода
Если хотя бы одна из оценок меньше 4, завершаем программу и печатаем 'YES' (пересдачи есть)
Если все пять оценок больше 3, печатаем 'NO' (студент закрыл сессию без пересдач)

In [27]:
```
i = 1
while i <= 5:
    note = int(input("Введите оценку: "))
    if note < 4:
        print('YES')
        break
    i += 1
else: # else находится на том же уровне отступа, что и while, поэтому относится имен
    print ('NO')
```

```
Введите оценку: 6
Введите оценку: 7
Введите оценку: 3
YES
```

Оператор **continue** позволяет сразу же перейти на новую итерацию цикла, не выполняя код, который написан внутри цикла ниже его.

Изменим условие задачи - теперь будем считать количество пересдач у студента

In [11]:
```
i = 1
retakes = 0
while i <= 5:
    note = int(input("Введите оценку: "))
    i += 1
    if note >= 4: # если пересдачи нет, сразу же идем проверять переменную i, без уб
        continue
    retakes += 1
print("Итого пересдач:", retakes)
```

```
Итого пересдач: 2
```

Операторами break и continue не стоит злоупотреблять, это может ухудшить читаемость кода.

Например, в предыдущем примере мы бы справились и без continue:

In [35]:
```
i = 1
retakes = 0
while i <= 5:
    note = int(input("Введите оценку: "))
    i += 1
    if note < 4:
        retakes += 1
print("Итого пересдач:", retakes)
```

```
Введите оценку: 5
Введите оценку: 6
Введите оценку: 7
Введите оценку: 1
Введите оценку: 7
Итого пересдач: 1
```

(∩ ` -´)⊃–☆˚.*･｡˚ Задача

Вася начал бегать и в первый день он пробежал X километров и выдохся. Вася поставил себе цель Y километров и решил узнать, когда он ее достигнет, если каждый день будет бегать дистанцию на 10% больше, чем в предыдущий.

Формат ввода

Программа получает на вход целые числа X, Y

Формат вывода

Одно целое число (день, когда Вася пробежит свою цель)

Примеры

Ввод:

10

21

Вывод:

9

In [13]:
```
x = int(input())
y = int(input())

day_count = 1
while x < y:
    x += x*0.1
    day_count += 1

print(day_count)
```

```
9
```

(∩ ` -´)⊃–☆˚.*･｡˚ Задача

Сложные проценты

Процентная ставка по вкладу составляет P процентов годовых, которые прибавляются к сумме вклада через год. Вклад составляет X рублей Y копеек. Дробное число копеек по истечении года отбрасывается. Выведите величину вклада в рубл

Формат ввода

Программа получает на вход целые числа P, X, Y, K.

Формат вывода

Программа должна вывести два числа: величину вклада через K лет в рублях и копейках. . Перерасчет суммы вклада (с отбрасыванием дробных частей копеек) происходит ежегодно.

Примеры
Тест 1
Входные данные:
12
179
0
5

Вывод программы:
315 43

Тест 2
Входные данные:
13
179
0
100

Вывод программы:
36360285 50

Тест 3
Входные данные:
1
1
0
1000

Вывод программы:
11881 92

In [61]:
```
# (∩ `-´)⊃━☆ﾟ.*･｡ﾟ

p = 12
x = 179
```

```python
y = 0
k = 5

year = 0 # заводим счетчик годов
end_year_amount = x * 100 + y # считаем начальную сумму в копейках

while year < k:
    percent = end_year_amount * p / 100 # считаем процент в этот год
    end_year_amount = int(end_year_amount + percent) # считаем итог в этот год
    year += 1 # Обновляем счетчик года

print(end_year_amount // 100, end_year_amount % 100)
```

```
315 43
```