Урок 3. Представление вещественных чисел в программе

Мы уже хорошо поработали с функциями и узнали про них довольно много. Теперь пришло время вернуться к работе с вещественными числами и заодно разобраться, как на самом деле десятичные дроби представляются в памяти компьютера. И для начала рассмотрим небольшую задачу.

Задача 1

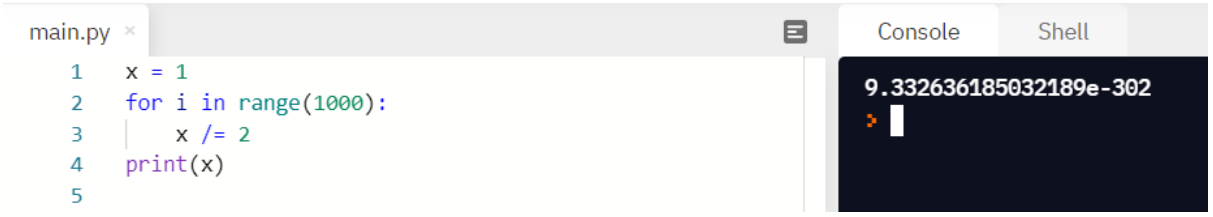
Один научно-исследовательский институт сейчас работает с очень большими и очень маленькими числами для вывода новых математических законов. Он дал нам задачу понять, какое число получится при делении единицы пополам 10 тысяч раз. Конечно же, недолго думая, мы просто написали для этого небольшую программку.

main.py

```
1   x = 1
2   for i in range(10000):
3       x /= 2
4   print(x)
5
```

Есть число, есть цикл, который повторяется 10 тысяч раз, а в конце мы выводим то, что получилось. Давайте проверим.

В результате мы видим 0. Давайте попробуем убрать один нолик из количества итераций и попробуем ещё раз.



Вот теперь какие-то числа выводятся. Пока непонятно, что это, но уже не ноль. А давайте заменим цикл **for** на цикл **while** и будем делить до тех пор, пока не будет ноль. И заодно заведём счётчик итераций и сам **x** будем выводить внутри тела цикла.

main.py

```
x = 1
count = 0
while x!= 0:
    x /= 2
    count +=1
    print(x)
print('Итераций:', count)

```

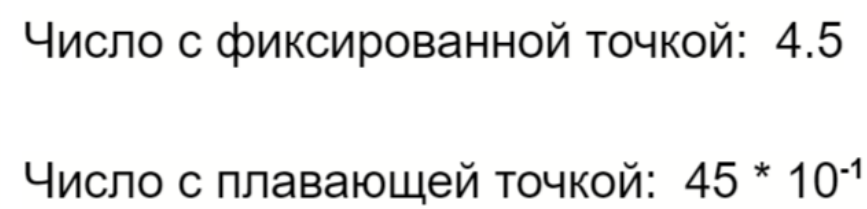
Тест:

```
4e-323
2e-323
1e-323
5e-324
0.0
Итераций: 1075
```

Ага, значит, максимум итераций у нас оказалось всего 1 075. Далековато до десяти тысяч! Но как же так? Чтобы понять, что произошло, давайте сначала разберёмся, что означают все эти записи в результате.

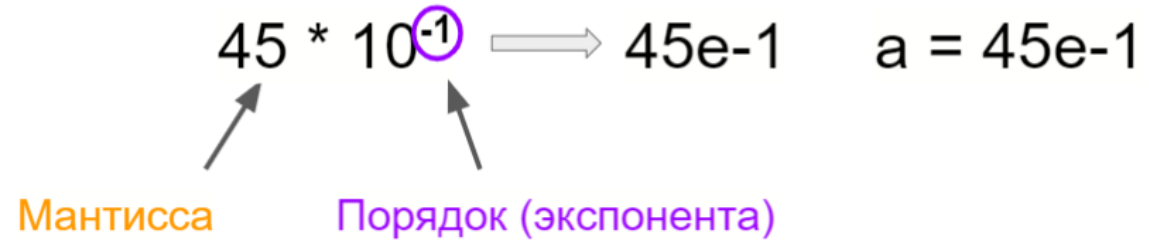
Представление чисел

Вообще, компьютеры могут работать только с целыми числами. А для работы с вещественными придумали хитрость: число хранится в памяти в виде двух целых.



Любую десятичную дробь можно представить в двух видах: классическом (с фиксированной точкой), например 4.5, и так называемом формате **плавающей точки**: - 45 * 10^(-1). В последнем случае наше итоговое число задаётся как некое целое

число, умноженное на какую-то степень основания нашей системы счисления (если вы не робот, скорее всего, вы тоже используете десятичную).



При этом число 45 называется **мантиссой** (эта часть непосредственно представляет наши цифры), а **порядок равен -1** — он записывается после буквы e, которая и обозначает степень десятки. Любое число может быть представлено в такой форме, и в подобном формате десятичные дроби хранятся в памяти компьютера. Именно экспоненциальную форму записи числа мы увидели в результате работы нашей программы.

Эта запись как раз хорошо отражает такой способ представления в компьютерах. В нашем примере мы можем записать это число как 45e-1. И точно так же можно записывать числа в самой программе. Перед буквой e идёт мантисса, а после неё — порядок.

```
450e-2     0.045e2
```

В таком представлении мы можем смещать десятичную точку влево или вправо по мантиссе, не забывая при этом менять порядок. Если мы смещаем точку на одну позицию вправо, то мы должны уменьшить порядок на единицу, и наоборот, смещая точку влево, мы должны увеличить порядок на один. Десятичная точка может как бы плавать, отсюда, кстати, и название чисел с плавающей точкой. Например, то же самое число можно записать как 450e-2 или 0.045e2.

```
4.5e0
```

Но вообще обычно делают так, **чтобы слева от точки стояла только одна цифра**, а всё остальное было записано справа. Это стандарт такой записи в программировании. То есть правильно было бы записать вот так: 4.5e0. Значит, число, которое мы получили в результате деления на 2, означает 5 * 10^(-324)!

Задача 2

Теперь давайте решим ещё одну задачу, чтобы закрепить работу с экспоненциальной формой числа. Предположим, учёные изучают различные планеты, чтобы понять,

какие из них могут быть пригодны для колонизации людьми. Но для начала им нужно знать базовые характеристики планеты, такие как плотность, массу и объём. У нас есть две планеты, объёмы которых известны заранее, а данные о массе поступают от других учёных. Нам нужно найти плотность каждой планеты и понять, какая из них ближе всего по плотности к нашей планете Земля. Плотность, кстати, это масса поделённая на объём.

Итак, нам изначально известны объёмы двух планет и плотность планеты Земля. Запишем их с помощью экспоненциальной формы записи числа.

main.py

```
1   vFirstPlanet = 1.43128e15  #км^3
2   vSecondPlanet = 6.254e13  #км^3
3   pEarth = 5.5153e12  #кг/км^3
```

Затем нам поступают данные о массе планет. Понятное дело, что масса планеты это далеко не самое маленькое число, значит, вводить мы её будем также с использованием буквы e. Тогда на вводе нам понадобится float.

```
5   mFirstPlanet = float(input('Масса первой планеты: '))
6   mSecondPlanet = float(input('Масса второй планеты: '))
```

Далее нам нужно рассчитать плотность каждой планеты. Мы уже знаем формулу: это масса поделить на объём. Поэтому создадим ещё две переменные и выведем наши плотности, чтобы хотя бы было понятно, что мы там насчитали. Затем проверим работу программы.

```
8   pFirstPlanet = mFirstPlanet / vFirstPlanet
9   pSecondPlanet = mSecondPlanet / vSecondPlanet
10  print('Плотность первой планеты:', pFirstPlanet)
11  print('Плотность второй планеты:', pSecondPlanet)
```

Тест:

```
Масса первой планеты: 1.8985e27
Масса второй планеты: 1.0243e26
Плотность первой планеты: 1326435079090.0452
Плотность второй планеты: 1637831787655.9004
>
```

Так, получились довольно большие числа. И на самом деле, так и должно быть, ведь плотность Земли тоже число немаленькое: у нас здесь аж 12-ая степень. И если мы посчитаем количество знаков в ответах до точки, то у нас как раз получится такая же степень. Нам осталось понять, какая из этих плотностей будет ближе к земной. Для этого мы можем просто сравнить разности, и если разность между земной плотностью

и первой будет меньше разности земной плотности и второй, то получится, что первая планета будет ближе.

```
13    if pEarth - pFirstPlanet < pEarth - pSecondPlanet:
```

Стоп, а если какая-нибудь плотность вообще будет больше земной? Это нам тоже нужно предусмотреть. Тогда просто возьмём модуль от каждой разности, а дальше выведем нужные сообщения.

```
13 if abs(pEarth - pFirstPlanet) < abs(pEarth - pSecondPlanet):
14     print('Плотность первой планеты ближе к плотности Земли')
15 else:
16     print('Плотность второй планеты ближе к плотности Земли')
17
```

Тест:

```
Масса первой планеты: 1.8985e27
Масса второй планеты: 1.0243e26
Плотность первой планеты: 1326435079090.0452
Плотность второй планеты: 1637831787655.9004
Плотность второй планеты ближе к плотности Земли
>
```

Вот так, с помощью экспоненциальной формы мы избавили себя от необходимости писать огромные числа вручную и записали их очень коротко и понятно.

Итоги урока

На этом наш урок завершается, поэтому пришло время подводить итоги. Что мы узнали:

- Любую десятичную дробь можно представить в формате плавающей точки. Она состоит из мантиссы, то есть цифр, числа и порядка.
- Такое число можно записывать внутри программы в экспоненциальной форме, где сначала идёт мантисса, затем буква e, которая заменяет десятку, а затем порядок.
- С помощью порядка мы можем сдвигать точку в нужную нам сторону. Поэтому эти числа и называются числами с плавающей точкой.