Программа Python.
2021-2022 год.

Тема: Среда программирования. Целые числа и строки.

Программа это текст.

Внутри текста две сущности.

Первая - это команды что сделать.

Вторая сущность - это данные.

Например, напишем следующие команды:

```
print(2)
print('Привет, Мир!')
```

В данном примере: функция print() это команда. Цифра - 2 или строка - Привет Мир! это данные. В первом случае (цифра 2) это числовые данные, во втором (строка – Привет, Мир!) это строковые данные.

Но иногда хочется, чтобы программа печатала разные значения, вычисляла значения, перед тем как вывести на печать.

Для этого на помощь приходят переменные.

```
m = 5
print(m)
```

теперь, если менять значение переменной m, программа будет печатать разные значения.

Переменные

Правило: Переменные хранят значения, а не выражения.

Простой пример выражения это комбинация значений данных, соединенных оператором.

Пример: **print(2*2+2)**

В скобках функции как раз находится выражение **2*2+2**

Порядок вычисления выражения происходит по правилам математики, но можно применять и скобки.

2*2+2 - называется выражением и возвращает единственное значение – ответ => 6!

Для выражения: 5 + будет ошибка.

Пример:

```
m = 5 + 5
print(m)
```

Результат: 10

```
m = 5 +
print(m)
```

Результат: Ошибка программы – (не хватает в выражении второго значения справа)

Объявление или создание переменной происходит с помощью инструкции присваивания - знак равно:

Пример: **mom = 7**

Чтобы посмотреть, что содержится в переменной, просто печатаем на экране ее имя – **print(mom).**

```
mom = 7
```

```
print(mom)
```

Для того, чтобы использовать переменную в программе, сначала ее надо объявить.

```
mom = 7
print(mom)
```

Результат: 7

```
print(mom)
mom = 7
```

Результат: Ошибка.

```
builtins.NameError: name 'mom' is not defined
```

Имя mom не определено! Язык Python ищет переменную mom в своей памяти для вывода на экран, с помощью функции print(), но не может найти.

В выражениях можно использовать переменные вместе с числовыми или текстовыми данными в комбинации:
Пример:

```
mom = 2
newMom = mom + 8
print(newMom)
```

Результат: 10

Значение переменной можно изменить только инструкцией присваивания:

```
mom = 7
print(mom)

mom + 3
print(mom)
```

Результат:
7
7 (не 10!)

Можно задать новое значение переменной mom с использованием ее старого значения:

```
mom = mom + 3
print(mom)
```

Результат: 10 (Теперь значение m поменялось!)

Переменные бывают разных типов.

Типы переменных в Python – это не что иное, как зарезервированные ячейки памяти для хранения значений. Это означает, что при создании переменной вы резервируете некоторое место в памяти.

Типы переменных в языке Питон не нуждаются в явном объявлении для резервирования места в памяти. Объявление происходит автоматически, когда вы присваиваете значение переменной.

Знак равенства (=) используется для присвоения типа переменным.

```
counter = 100          # Присвоение целого числа
miles   = 1000.0       # Число с плавающей точкой
name    = "Джон"       # Строка
```

Давайте посмотрим какой тип у каждой из этих переменных.

```
print(type(counter))
print(type(miles))
print(type(name))
```

<class 'int'>

```
<class 'float'>
<class 'str'>
```

Но переменные также для языка Python являются объектами.
И эти объекты откосятся к сущности: **данные (не команды)!**

Python имеет шесть основных типов данных:

Числа - 1, 2, 3, 1.12, 10.33 (integer (int), float)

Строка - 'hostname'

Список - [1, 2, 3, 4]

Логический тип данных - тип Bool (True, False)

Кортеж (список из строк) - Кортеж в Python это:
последовательность элементов, которые разделены между собой запятой и заключены в скобки.
list_keys = ['hostname', 'location', 'vendor', 'model'].
Грубо говоря, кортеж - это список, который нельзя изменить.
То есть, в кортеже есть только права на чтение.

Словарь - {'Name': 'John', 'Age': 29, 'Company': 'GOOGLE'}

Словарь в Python – это тип данных, который может имитировать реальную организацию данных, в которой существует определенное значение для определенного ключа.

Это изменяемая структура данных.

Словарь определяется в элементах ключи и значения:

Ключи должны быть единым элементом.

Значение может быть любого типа, например, список, кортеж, целое число и т. д.

Зачем типы?

Значение переменной нам необходимо хранить на компьютере. Поэтому типы данных определяют метод хранения для каждого из них.

Повторяю - метод хранения определяется автоматически Python. Нам достаточно только объявить имя переменной и присвоить ей определенное значение. По этому значению и будет определяться автоматически тип данных.

Как мы уже выяснили, компьютер с помощью процессора может понимать только 0 и 1. И поэтому в ячейки своей памяти он сможет записывать 0 и 1.

Логично предположить, что язык Python хорошо умеет переводить любые числа или строки в 0 и 1 для отправки процессору компьютера на хранение.

Когда Python переводит данные в вид 0 и 1, он придерживается определенных правил, перед тем как сохранять числа или строки в ячейки памяти?

Давайте посмотрим, как Python будет хранить большие числа.

Например: число 2021**2021

Можно перейти в Python Shell, чтобы не писать print(), и сразу получить ответ. Консоль это позволяет.

20212021**
получаем число

получим количество знаков этого числа:

```
print(len(str(2021**2021)))
```

6681

Тут Python ведет себя хорошо и выделяет для хранения такого большого числа необходимое место на компьютере.

А теперь посмотрим, как Python сохранит бесконечно длинное не целое число.

Пример:

```
x = 1
y = 3

x/y
```

ответ: 0.3333333333333333

ответ не бесконечное число!!!!

Поэтому запоминаем правило: Все вычисления с не целыми числами могут нести в себе ошибку. Потому, что происходит округление результата!

1 000.000 - миллион
1 000.000.000 - миллиард
1 000.000.000.000 - триллион

Еще один пример, можно уже перейти в программу, выйти из консоли:

```
x = 1
print(x / 100000000000000)
```
- (14 нулей - сто триллионов)

1e-14 => 0.00000000000001

Python перед выводом очень маленького числа (**0.00000000000001**) перевел его в экспоненциальный вид.

Немного ниже мы коснемся правил перевода чисел в экспоненциальный вид.

А пока еще пример:

Умножим наше число на делитель:

```
print(x / 100000000000000 * 100000000000000)
```

```
1.0
```

Получаем единицу. Пока все правильно!

А если еще добавить столько же нулей к делителю?

```
print(x / 10000000000000000000000000000 *
10000000000000000000000000000)
```

```
0.9999999999999999
```

уже не 1! А это уже математически неправильно!

И произошло это по причине того, что Python имеет свойство обрезать не целые числа!

Еще пример:

```
x = 2
print((x ** (0.5))**2)
```

```
2.0000000000000004
```

уже не 2!

Вывод:

Если числа целые, то Python может их хранить в памяти точно. Если числа не целые (действительные), то Python их немного обрезает.

Поэтому операции с не целыми числами могут содержать неточность!

Так что пока мы будем работать с целыми числами.

Как компьютер будет хранить некоторые большие числа для экономии памяти: Компьютер будет переводить число в экспоненциальный вид, если это возможно, для сокращения хранения символов в ячейке памяти.

Пример:

```
x = 11111111111111111111111111111111 (33 единицы)
y = 2
print(x / y)
```

```
5.555555555555556e+31
```

На компьютере (в частности в тексте компьютерных программ) экспоненциальную запись записывают в виде МЕр (пример 1e-10), где: М — мантисса,
*Е (exponent) — буква Е в числе, означающая «*10^» («…умножить на десять в степени…»),*
р — порядок степени.
Это необходимо для представления очень больших и очень малых чисел, а также для унификации их написания.

Добавим в нашу программу дополнительную переменную z:

```
z = 5.555555555555556 * (10**31)
print(z)
```

```
5.555555555555556 * (10**31)
первая часть - мантисса
вторая часть - экспонента
```

Запускаем полностью программу:

```
x = 11111111111111111111111111111111
y = 2
print(x / y)
z = 5.555555555555556 * (10**31)
print(z)
```

Результат:

```
5.555555555555556e+31
5.555555555555556e+31
```

Еще один пример: Типы числовых данных - int и float

```
x = 1.2
y = 12
print('x -', type(x))
print('y -', type(y))
```

Результат:

```
x - <class 'float'>
y - <class 'int'>
```

Целые числа обозначаются в языке Python **- int.** Это сокращение от английского слова **- integer (целое число).**

Не целые числа, числа, у которых есть знаки после запятой (в Python запятая не пишется, вместо запятой мы будем писать точку), обозначаются **- float.** От английского слова **float (плыть).** Точка может перемещаться по числу (влево, вправо), в зависимости от знаков после запятой (десятки, сотни, тысячные после запятой и т.д.)

Если происходит деление, то всегда возвращает тип значения float, даже если числа, участвующие в делении целые (тип int)

Пример:

```
x = 10
y = 5
print(type(x))
print(type(y))
print(x / y)
print(type(x / y))
```

Результат:

```
<class 'int'>
<class 'int'>
2.0
<class 'float'>
```

Рассмотрим еще одну функцию, встроенную в Python - abs():

```
x = -10
y = abs(x)
print(x, y)
```

```
-10 10
```

где abs() - это функция, которая вычисляет модуль числа - абсолютное его значение.

Итак - целые и не целые числа мы прошли. Целые числа обозначаются в языке Python - int. Это сокращение от английского слова - integer (целое число). Не целые числа обозначаются в языке Python - float. Это от английского слова - float (плыть).

Теперь пришло время поговорить о строках (или о типе данных, как строка):

Как в языке задать строку?

Если написать:
x = abc, то будет ошибка. abc будет языку казаться именем переменной abc, но она не определена в начале программы, чтобы можно было ее использовать.

Поэтому необходимо использовать одинарные или двойные кавычки.
x = "abc"

Как хранится строка в памяти компьютера?

Логично, что строка должна быть переведена в вид 0 и 1 - этим пусть займется сам Python.

Каждой букве присваивается свое цифровое значение 0 и 1, а эти цифровые значения уже хранятся отдельно в ячейках памяти компьютера.

Практика: Есть любое целое число. Задача напечатать последнюю цифру числа.

Идея: Остаток от деления числа на 10 и будет последней цифрой любого целого числа.

```
x = 553636377838838
print(x%10)
```
ответ - 8