**Информация об уроке**

**Модуль: 1**

**Номер занятия: 6**

**Тип занятия: Лекция**

**Название занятия/ Тема занятия:** Переменные, типы данных и операторы

**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. **Переменный**
2. **Типы данных, преобразование типа данных;**
3. **Операторы.**

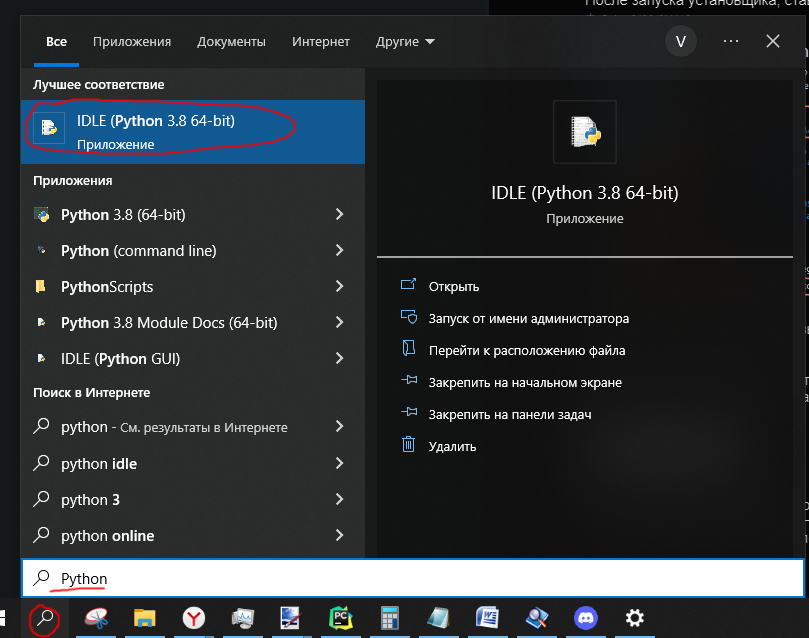
Эти темы являются основополагающими не только в Python, но и в целом во всех языках программирования. Поняв логику работы программ, перейти с одного языка на другой достаточно просто, ведь всё изучение будет упираться в отличия синтаксиса между разными языками программирования.

1. **Актуализация**

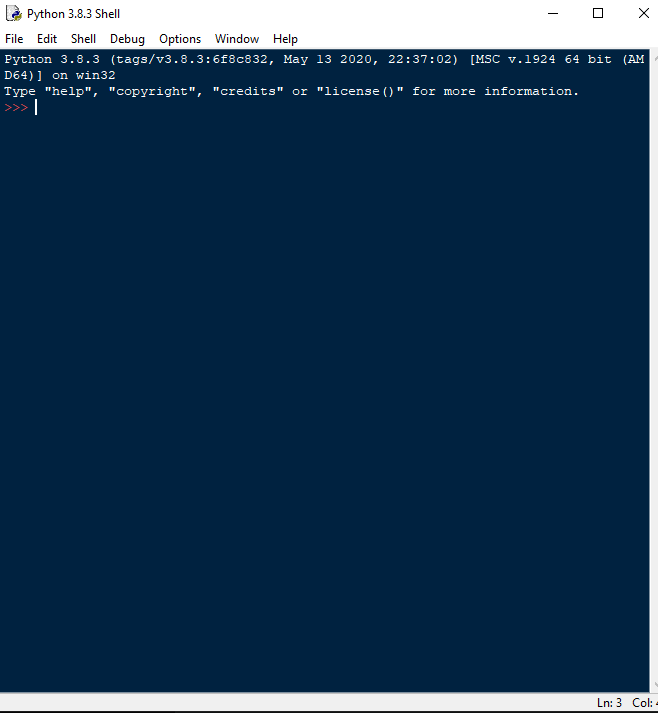
**Инструкция** - некоторое действие. Оно может означать математическую операцию, сравнение или сохранение / удаление данных из переменной. Программы состоят из последовательного набора таких инструкций (действий).

**Алгоритм** – последовательная совокупность заданных инструкций, которая приводит к решению задачи.

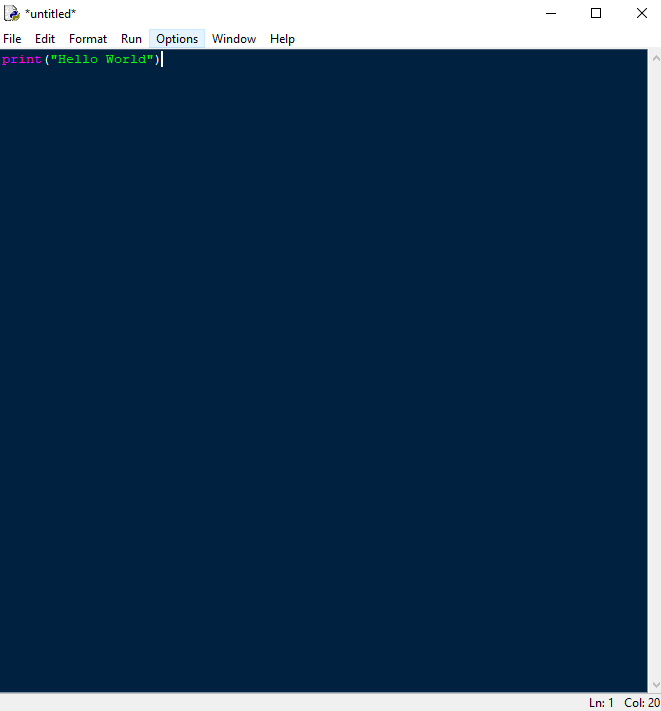
Ищем в поиске windows Python:



Только что мы запустили консоль Python:



Для создания программы, нажимаем слева сверху на file - new file:



Все программы, мы будем писать здесь.

Программа на языке Python состоит из набора инструкций. Каждая инструкция помещается на новую строку.

Большую роль в Python играют отступы. Неправильно поставленный отступ фактически является ошибкой, поэтому стоит помещать новые инструкции сначала строки.

Python - регистрозависимый язык, поэтому выражения print и Print или PRINT представляют разные выражения.

Для отметки, что делает тот или иной участок кода, применяются комментарии. При трансляции и выполнении программы интерпретатор игнорирует комментарии, поэтому они не оказывают никакого влияния на работу программы. Комментарии в Python бывают блочные и строчные и ставятся с помощью символа

Строчные комментарии предваряются знаком решетки - #. Они могут располагаться на отдельной строке.

В блочных комментариях до и после текста комментария ставятся три одинарные кавычки: '''текст комментария'''. Например:

Для вывода информации на консоль предназначена встроенная функция **print()**. При вызове этой функции ей в скобках передается выводимое значение.

Например, мы хотим, чтобы все значения выводились на одной строке. Для этого нам надо настроить поведение функции с помощью параметра **end**. Этот параметр задает символы, которые добавляются в конце к выводимой строке. При применении параметра **end** вызов функции **print()** выглядит следующим образом:



По умолчанию end равен символу "\n", который задает перевод на следующую строку. Собственно поэтому функция print по умолчанию выводит передаваемое ей значение на отдельной строке.

Наряду с выводом на консоль мы можем получать ввод пользователя с консоли, получать вводимые данные. Для этого в Python определена функция **input()**. В эту функцию передается приглашение к вводу. А результат ввода мы можем сохранить в переменную.

Стоит учитывать, что все введенные значения рассматриваются как значения типа str, то есть строки. И даже если мы вводим число, как в втором случае в коде выше, то Python все равно будет рассматривать введенное значение как строку, а не как число.

1. **Основное содержание**

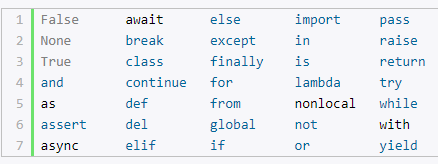
В программировании часто мы используем информацию, какие-либо данные и операции, применимые к ним. Представим ситуацию, когда нам необходимо анализировать результат операций между тысячей цифр. Отсюда возникает вопрос: можно ли как-то запомнить эти значения для повторного использования в программе?

Ответ - да. Во всех наших последующих программах, результаты операций мы будем сохранять. Для этого в языках программирования есть переменные.

**1. Переменные.**

**Переменные**

Переменные предназначены для хранения данных. Название переменной в Python должно начинаться с алфавитного символа или со знака подчеркивания и может содержать алфавитно-цифровые символы и знак подчеркивания. И кроме того, название переменной не должно совпадать с названием ключевых слов языка Python. Ключевых слов не так много, их легко запомнить:



создадим переменную:



Здесь определена переменная **name**, которая хранит строку **"Tom"**.

В Python применяется два типа наименования переменных: **camel case** и **underscore notation**.

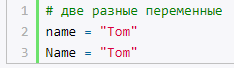
**Camel case** подразумевает, что каждое новое подслово в наименовании переменной начинается с большой буквы. Например:



**Underscore notation** подразумевает, что подслова в наименовании переменной разделяются знаком подчеркивания. Например:

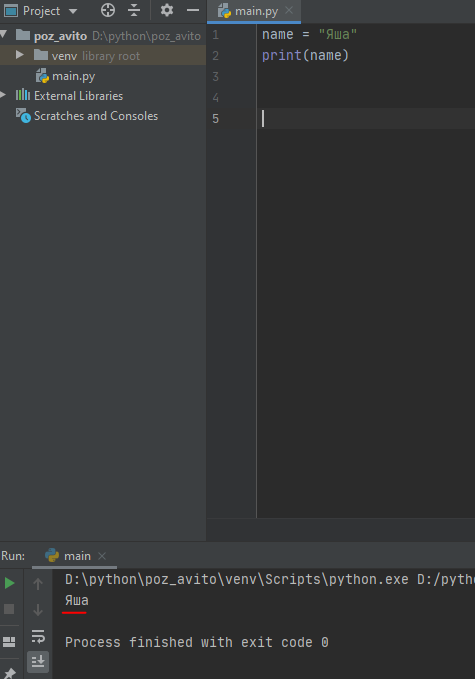


И также надо учитывать регистрозависимость, поэтому переменные name и Name будут представлять разные объекты.

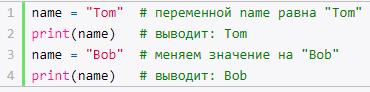


Определив переменную, мы можем использовать её в программе. Например, попытаться вывести ее содержимое на консоль с помощью встроенной функции print:





Отличительной особенностью переменной является то, что мы можем менять ее значение в течение работы программы:



**1. Типы данных.**

Компьютеру необходимо заранее знать какого рода данные (тип данных) будут храниться в той или иной области памяти (переменной) - целое число, дробное число, логическая функция, текст или комплексное число. Это нужно для того, чтобы ЭВМ понимала какие операции и действия мы можем совершать над переменной.

Представим ситуацию, что в двух переменных хранятся разные типы данных: строка и число. В программе встречается инструкция – сложение двух этих переменных. Когда интерпретатор дойдёт до этой строчки, программа аварийно завершится, поскольку не только компьютер не знает как складывать строки и числа, но и человек. Самостоятел

**Тип данных** – описание того, какого рода данные хранятся в переменной.

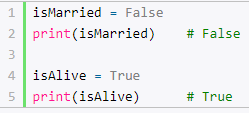
Переменная хранит данные одного из типов данных:

* **int** (integer) – число
* **float** (плавающая точка) – дробное число
* **str** (string) – строка
* **bool** (булева функция) – True или False (правда или ложь / из двоичной логики)

Это конечно же не все типы данных, которые существуют в языке Python, однако они являются базовыми.

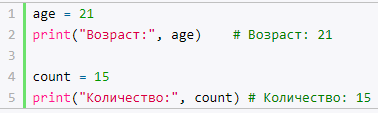
**Логические значения**

Тип **bool** представляет два логических значения: **True** (верно, истина) или **False** (неверно, ложь). Значение **True** служит для того, чтобы показать, что что-то истинно. Тогда как значение **False**, наоборот, показывает, что что-то ложно. Пример переменных данного типа:



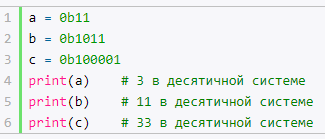
**Целые числа**

Тип **int** представляет целое число, например, 1, 4, 8, 50. Пример:

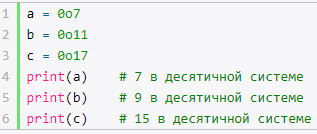


По умолчанию стандартные числа расцениваются как числа в **десятичной системе**. Но Python также поддерживает числа в двоичной, восьмеричной и шестнадцатеричной системах.

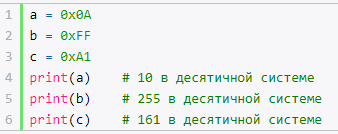
Для указания, что число представляет **двоичную систему**, перед числом ставится префикс **0b**:



Для указания, что число представляет **восьмеричную систему**, перед числом ставится префикс **0o**:



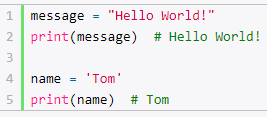
Для указания, что число представляет **шестнадцатеричную систему**, перед числом ставится префикс **0x**:



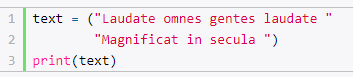
Стоит отметить, что в какой-бы системе мы не передали число в функцию **print** для вывода на консоль, оно по умолчанию будет выводиться **в десятичной системе**.

**Строки**

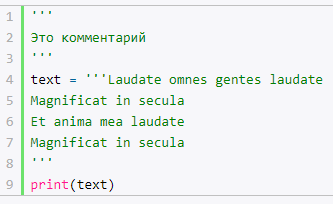
Тип **str** представляет **строки**. Строка представляет последовательность символов, заключенную в одинарные или двойные кавычки, например "hello" и 'hello'. В Python 3.x строки представляют набор символов в кодировке Unicode



При этом если строка имеет много символов, ее можем разбить ее на части и разместить их на разных строках кода. В этом случае вся строка заключается в круглые скобки, а ее отдельные части - в кавычки:



Если же мы хотим определить многострочный текст, то такой текст заключается в тройные двойные или одинарные кавычки:



При использовании тройных одинарных кавычек не стоит путать их с комментариями: если текст в тройных одинарных кавычках присваивается переменной, то это строка, а не комментарий.

**Управляющие последовательности в строке**

Строка может содержать ряд специальных символов - **управляющих последовательностей**. Некоторые из них:

* **\** - позволяет добавить внутрь строки слеш
* **\' -**  позволяет добавить внутрь строки одинарную кавычку
* **\" -**  позволяет добавить внутрь строки двойную кавычку
* **\n -**  осуществляет переход на новую строку
* **\t -**  добавляет табуляцию (4 отступа)

Применим несколько последовательностей:



Консольный вывод:



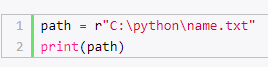
Хотя подобные последовательности могут нам помочь в некоторых делах, например, поместить в строку кавычку, сделать табуляцию, перенос на другую строку. Но они также могут и мешать. Например:



Здесь переменная path содержит некоторый путь к файлу. Однако внутри строки встречаются символы "\n", которые будут интерпретированы как управляющая последовательность. Так, мы получим следующий консольный вывод:



Чтобы избежать подобной ситуации, перед строкой ставится символ **r**



**Вставка значений в строку**

Python позволяет встравивать в строку значения других переменных. Для этого внутри строки переменные размещаются в фигурных скобках **{}**, а перед всей строкой ставится символ **f**:  

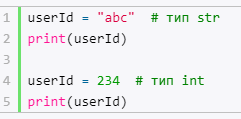

В данном случае на место {userName} будет вставляться значение переменной userName. Аналогично на вместо {userAge} будет вставляться значение переменной userAge.

**Динамическая типизация**

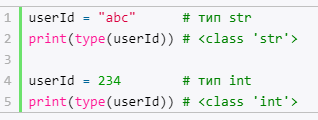
Python является языком с динамической типизацией. А это значит, что переменная не привязана жестко с определенному типу.

Тип переменной определяется исходя из значения, которое ей присвоено. Так, при присвоении строки в двойных или одинарных кавычках переменная имеет тип str. При присвоении целого числа Python автоматически определяет тип переменной как int. Чтобы определить переменную как объект float, ей присваивается дробное число, в котором разделителем целой и дробной части является точка.

При этом в процессе работы программы мы можем изменить тип переменной, присвоив ей значение другого типа:



С помощью встроенной функции **type()** динамически можно узнать текущий тип переменной:



**Преобразование типа данных**

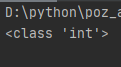
Для изменения типа данных используется следующая конструкция:



Пример:



Консольный вывод:



Выше было написано, что при вводе через консоль: a = input(), Python воспринимает введенную последовательность символов как **str.** Чтобы Python сразу воспринимал ввод как конкретный тип данных, можно написать так:

a = **int(**input()**)** или a = **float(**input()**)**

В этих случаях введенная последовательность символов будет сразу числовым типом данных.

Промежуточный итог по части:

* Переменные предназначены для хранения данных.
* **Camel case** подразумевает, что каждое новое подслово в наименовании переменной начинается с большой буквы
* **Underscore notation** подразумевает, что подслова в наименовании переменной разделяются знаком подчеркивания
* Определив переменную, мы можем использовать её в программе
* Значение переменной можно менять в процессе работы программы
* **Тип данных** – описание того, какого рода данные хранятся в переменной
* Переменная хранит данные одного из типов данных:
  + **int** (integer) – число
  + **float** (плавающая точка) – дробное число
  + **str** (string) – строка
  + **bool** (булева функция) – True или False (правда или ложь / из двоичной логики)
* Объявление (создание) переменных разных типов данных:

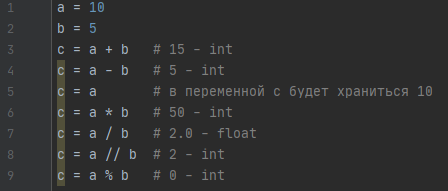
| **int** | **float** | **str** | **Bool** |
| --- | --- | --- | --- |
| a = 10 | a = 3.5 | a = “Hello World!” | a = True |

**3. Операторы.**

Python поддерживает все распространенные арифметические операции:

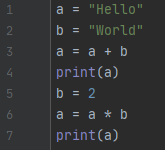
* + сложение
* - вычитание
* = присваивание
* \* умножение
* / деление
* \*\* возведение в степень
* // целочисленное деление
* % остаток от деления

Пример операций на двух **целочисленных** **(int)** переменных:

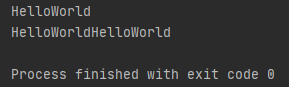


**ПРИМЕЧАНИЕ!**

Также возможно сложение строк: **str + str**; и умножение строки на число: **str \* int**:



Консольный вывод:



**Операции и их направление**

**\*\*** - Справа налево

**\* / // %** - Слева направо

**+ -** - Слева направо



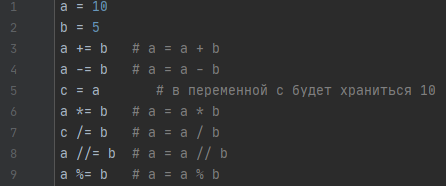
Здесь начале выполняется возведение в степень (5 \*\* 2) как операция с большим приоритетом, далее результат умножается на 4 (25 \* 4), затем происходит сложение (3 + 100) и далее опять идет сложение (103 + 7).

Чтобы переопределить порядок операций, можно использовать скобки:



**Арифметические операции с присвоением**

Ряд специальных операций позволяют использовать присвоить результат операции первому операнду:



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.