Лабораторная работа №1

Переменные и арифметические операции

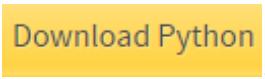
Ввод и вывод информации, комментарии

Если программа не установлена на компьютер, то:

Зайдите в компьютер, используя свой логин и пароль.

Зайдите в папку Документы и создайте папку Python и папку PythonLab.

Откройте окно браузера и в поисковой строке наберите слово “Python”, перейдите по ссылке на сайт python.org.

Выберите вкладку Downloads и установите последнюю версию программы, щелкнув на кнопку 

.

Запустите файл установки.

В открывшемся окне уберите галочку с надписи Install launcher for all users (recommended), а в надписи Add Python 3.8 to PATH галочку поставьте.

Выберите Customize Installation, нажмите Next, далее выберите Browse и найдите свою папку Python, Install, Close.

С помощью кнопки  запустите список приложений и найдите программу IDLE (Python 3.8 32-bit).

Нажав на программу правой клавишей мыши, в появившемся меню выберите «Закрепить на панели задач».

Войдите в программу, нажав один раз на иконку на панели задач.

Если программа установлена, то:

Найдите в поиске и запустите программу IDLE (Python 3.8 32-bit) или запустите ее с помощью кнопки  на панели задач.

У вас открылось окно, в котором вы будете видеть результат своего труда.

Выбираете ***File*** – ***New file***, открылось окно, в котором вы будете работать и писать программу.

С помощью функции или инструкции `print()` информация выводится на консоль. При этом следует помнить о том, что каждый код помещается в новую строку без отступа, если он не является условной конструкцией другого кода, также Python является регистрозависимым языком.

В открывшемся окне наберите следующий скрипт:

```
print ("Привет, мир!")
print ("Привет вновь!")
print ("Мне нравится это печатать…")
print ("Ура! Я печатаю!")
print ('Это весело!')
print ("Это занятие как 'волшебство'.")
print ('Я бы "сказал", что это круто!')
```

Выберите ***File*** – ***Save as***, выберите папку PythonLab и сохраните документ под названием Ex1.1.

Запустите программу, выполнив ***Run-Run Module*** или нажав F5.

Добавьте в ваш сценарий еще одну строку текста, которую Python должен вывести с помощью функции `print`.

Если же нам необходимо вывести несколько значений на консоль, то мы можем передать их в функцию `print` через запятую. Проверьте это, введя следующий скрипт:

```
print("Замечательное утро:", "Кофе", "и", "круассан")
```

У функции `print` есть параметр `sep`, который позволяет задать разделитель между выводимыми значениями.

```
print(5,7,9)
print(1,8)
print(6)
```

После вывода на экран данного скрипта, Вы увидите, что все числа выводятся в строчку через пробел. Чтобы заменить пробел на какой-нибудь символ, необходимо использовать параметр sep.

```
print(5,7,9,sep=",")#числа выводятся в строчку через запятую
print(1,8,sep="&")#числа выводятся в строчку через амперсанд
print(6,sep="/")#ничего не выводится, так как разделитель может стоять минимум между 2 значениями
```

У функции print есть еще один параметр end, который отвечает за то, что будет выводиться после всех элементов в конце строки.

По умолчанию данный параметр имеет значение end="\n", который задает переход на новую строку.

```
print(5,7,9,sep=",",end="  ")#вместо перехода на новую строку следующие значения выводятся на экран через пробел
print(1,8,sep="&")
print(6,sep="/")
```

Комментарии в коде программ очень важны. Они используются, чтобы на человеческом языке сообщить какую-либо информацию, а также для отключения (закомментирования) части кода вашей программы, если нужно, чтобы временно он не работал. Рассмотрим, как комментарии используются в языке Python. Добавьте в ваш документ следующий сценарий кода:

```
# Комментарии позволяют разобраться в предназначении кода.
# Строки после символа # игнорируются python.
print("Это моя программа.") # и этот комментарий игнорируется.
# С помощью комментариев можно "отключать" части кода:
```

```
# print ("Код не работает.")
print("Код работает.")
print("И это # гениально")
```

Комментарии бывают *блочные*, они ставятся в начале строки, и *строчные*, располагаются на той же строке, что инструкции языка.

Символ решетки также называют *октоторпом*, *хешем*, *знаком номера* и другими именами, выберите то, которое вам по душе.

Если символ # находится внутри строки, перед заключительной кавычкой ", этот символ решетки отображается обычным текстом и не считается началом комментария. Если вам нужен комментарий из нескольких строк, укажите символ # в начале каждой строки.

Если функция `print` отвечает за вывод информации на консоль, то функция `input` отвечает за ввод, то есть она берет какие – то значения от пользователя.

```
input("Как твои дела?")
```

В консоли введите ответ на вопрос и нажмите Enter. Удалите эту команду.

Далее введите следующие коды:

```
input("Ты сегодня завтракал?")
input("Выпьем вместе кофе?")
```

Как вы считаете, почему вторая фраза не появилась на экране?

В качестве необязательного параметра эта функция принимает приглашение к вводу и возвращает введенную строку, которую мы можем сохранить в переменную:

```
name = input("Введите имя: ")
print("Привет", name)
```

Переменные и типы данных

Переменная – это именованная ячейка памяти, в которую помещается какая – либо информация, откуда, в случае необходимости, мы эту информацию можем извлечь.

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: `and`, `as`, `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `False`, `finally`, `for`, `from`, `global`, `if`, `import`, `in`, `is`, `lambda`, `None`, `nonlocal`, `not`, `or`, `pass`, `raise`, `return`, `True`, `try`, `while`, `with`, `yield`. Эти слова запоминать не обязательно, ведь при написании скрипта они подсвечиваются цветом.

В Python применяется два типа наименования переменных: *camel case* (каждое новое подслово в наименовании переменной начинается с большой буквы) и *underscore notation* (подслова в наименовании переменной разделяются знаком подчеркивания). Например,

```
userName = "Tom" # Camel case
user_name = "Tom" # Underscore notation
```

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

Переменная хранит данные одного из типов данных. В Python существует множество различных типов данных, которые подразделяются на категории: числа, последовательности, словари, наборы:

boolean - логическое значение True или False

int - представляет целое число, для хранения которого использует 4 байта в памяти компьютера.

float - представляет число с плавающей точкой, для хранения которого используется 8 байт, например, 1.2 или 34.76

complex - комплексные числа

str - строки, например "hello". В Python 3.x строки представляют набор символов в кодировке Unicode

bytes - последовательность чисел в диапазоне 0-255

byte array - массив байтов, аналогичен bytes с тем отличием, что может изменяться

list - список

tuple - кортеж

set - неупорядоченная коллекция уникальных объектов

frozen set - то же самое, что и **set**, только не может изменяться (immutable)

dict - словарь, где каждый элемент имеет ключ и значение

Python является языком с *динамической типизацией*. Он определяет тип данных переменной 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. Число с плавающей точкой можно определять в экспоненциальной записи:

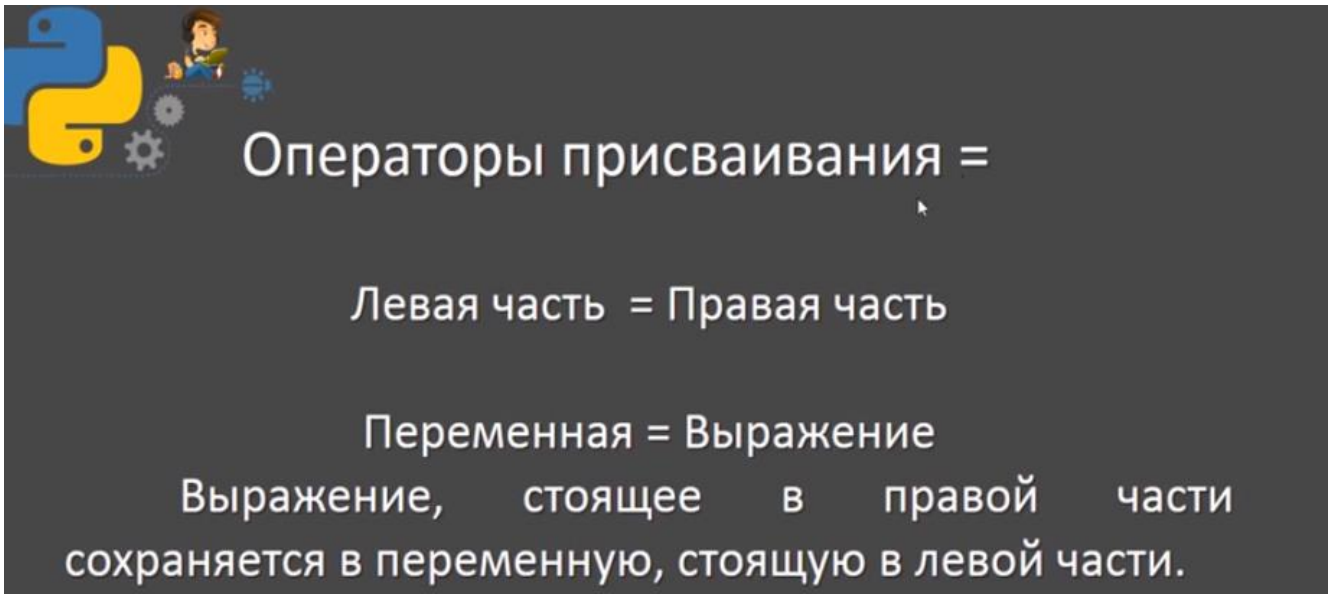
```
x = 3.9e3
print(x)  # 3900.0
 x = 3.9e-3
print(x)  # 0.0039
```

Число `float` может иметь только 18 значимых символов. Так, в данном случае используются только два символа - 3.9. И если число слишком велико или слишком мало, то мы можем записывать число в подобной нотации, используя экспоненту. Число после экспоненты указывает степень числа 10, на которое надо умножить основное число - 3.9.

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

```
a = 4 #создали новую переменную и данной переменной присвоили значение типа int
print(a)
a = 5.4 #переменная осталась та же, но этой переменной мы присвоили значение нового типа float, заменив старые данные типа int
print(a)
a = "Hi"#снова заменили содержание переменной a
print(a)
```

Оператор присваивания обозначается значком =. У данного оператора всегда есть правая и левая части. Оператор присваивания сначала вычисляет все, что стоит в правой части, а затем получившиеся данные сохраняет в левую часть. Если при этом, в левой части уже было сохранено какое – то значение, оно заменится на новое.



Прежде чем проверять в программе следующий скрипт, подумайте, **какое число программа выведет на экран**?

```
a = 5
b = 10
b = b*2
c = a + b
print(c)
```

Если у вас получился ответ 15 – это неправильный ответ, так как в третьей строке с помощью команды `b = b*2` программа записала в переменную `b` новое значение, а именно `10*2`, то есть `20`. Соответственно, в переменной `c` мы складывали уже не `5` и `10`, а `5` и `20`.

Также в Python вы можете выполнять массовые и множественные присвоения переменных.

```
a = b = c = 4
print(a)
print(b)
print(c)
m,n = 3,7
print(m)
print(n)
```

Обратим еще раз наше внимание на вывод информации на экран.

```
rubles = 100
kop = 99
print("У меня rubles рублей kop копеек")#программа вывела на экран всю строку полностью

rubles = 100
kop = 99
print("У меня", rubles, "рублей", kop, "копеек") #все слова берем в кавычки, переменные прописываем без кавычек, после каждого значения ставим запятую

rubles = 100
kop = 99
print("У меня %s рублей %s копеек"%(rubles,kop)) #специальный символ %s прописываем в том месте, куда
```

необходимо подставить значения, а затем в конце строки перечисляем данные значения в порядке появления

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

```
user_id = "12tomsmith438"
print(type(user_id))

user_id = 234
print(type(user_id))
```

Убедитесь, что программа верно определила тип данных в обоих случаях.

Числа и математика

Python поддерживает все распространенные арифметические операции:

+	сложение двух чисел
-	вычитание двух чисел
*	умножение двух чисел
/	деление двух чисел
//	целочисленное деление, возвращает целочисленный результат деления, отбрасывая дробную часть
**	возведение в степень
%	получение остатка от деления

При последовательном использовании нескольких арифметических операций их выполнение производится в соответствии с их приоритетом. В начале выполняются операции с большим приоритетом. Приоритеты операций в порядке убывания приведены ниже.

Высший приоритет: **

Средний приоритет: * / % //

Низкий приоритет: + -

Ряд специальных операций позволяют присвоить результат операции первому операнду.

+=	присвоение результата сложения
-=	присвоение результата вычитания
*=	присвоение результата умножения
/=	присвоение результата от деления
//=	присвоение результата целочисленного деления
**=	присвоение степени числа
%=	присвоение остатка от деления

1) Создайте новый файл Ex1.2.

2) Введите следующие коды:

```
test = 12 # введение новой переменной
print(test) # выводим на экран число

num_1 = input("Введите первое число: ") #просим пользователя ввести любое число, и это число будет первой переменной
num_2 = input("Введите второе число: ") #вводим вторую переменную
res = num_1 + num_2 # вводим третью переменную
print("Сумма этих двух чисел равна:", res)
```

Как вы считаете, почему действие сложения не выполнилось, а числа просто записались в строку?

Дело в том, что когда мы вводим информацию с клавиатуры, программа распознает её как `string`, то есть как строку, а не как числа.

Для того, чтобы программа поняла, что мы вводим именно числа, изменим наш сценарий следующим образом:

```
num_1 = int(input("Введите первое целое число: ")) #int представляет целое число
num_2 = int(input("Введите второе целое число: ")) #вводим вторую переменную
```

```
res = num_1 + num_2 # вводим третью переменную
print("Сумма этих двух чисел равна:", res)
```

Также можно задать тип `float` или указать тип данной в другом месте, как это показано в примере ниже:

```
num_1 = input("Введите первое целое число: ")
num_2 = int(input("Введите второе целое число: ")) #вводим вторую переменную
res = int(num_1) + num_2 # вводим третью переменную
print("Сумма этих двух чисел равна:", res)
```

Уберите тип `int` в одной из строк и запустите выполнение. Что у вас получилось?

Снова пропишите `int` и внесите следующие изменения в сценарий:

```
num_1 = input("Введите первое целое число: ")
num_2 = int(input("Введите второе целое число: ")) #вводим вторую переменную
res = int(num_1) + num_2 # вводим третью переменную
del res
print("Сумма этих двух чисел равна:", res)
```

Как вы считаете, почему программа выдала ошибку? Какое действие выполняет команда `del`?

```
num_1 = float(input("Введите первое число: "))
num_2 = float(input("Введите второе число: "))
res = num_1 + num_2
Res *= 3 #присваивает действие умножения аналогично действию res = res*3
```

```
print("Результат всех действий равен:", res)
```

Проверьте выполнение кода, почему результат не умножился на 3? Исправьте ошибку и снова выполните запуск.

Просмотрите следующую сценарий и, не вводя его в программу, посчитайте конечный результат.

Проверьте свои вычисления с помощью Python.

```
num_1 = 3
num_2 = 2
num_3 = 50
res = num_2+num_3//num_1**num_2%num_1
print("Результат всех действий равен:", res)
```

Если ваши результаты не совпали, найдите ошибку в своих рассуждениях.

Рассмотрите еще один пример.

```
first_number = 2.0025
second_number = 0.2468
third_number = first_number + second_number
print(round(third_number,    3))         #округление результата
```

Первый параметр функции - округляемое число, а второй - сколько знаков после запятой должно содержать получаемое число.

Обратим наше внимание на использование операций // (деление нацело) и % (получение остатка от деления). Данные арифметические действия позволяют решить ряд задач.

Дано целое положительное число, наша задача вывести его последнюю цифру.

```
a = int(input())
print(a%10)
```

Дано целое положительное число, наша задача вывести разряд сотен этого числа.

```
a = int(input())
print(a//100%10)
```

Дано целое положительное трехзначное число, ваша задача найти сумму разрядов этого числа.

```
a = int(input())
print(a//100%10+a//10%10+a%10)
```

n школьников делят k яблок поровну, неделяющийся остаток остается в корзинке. Сколько яблок достанется каждому школьнику? Программа получает на вход сперва число n, а потом k.

```
n = int(input())
k = int(input())
print(k//n)
```

Дано число n. С начала суток прошло n минут. Определите, сколько часов и минут будут показывать электронные часы в этот момент.

Программа должна вывести два числа: количество часов (от 0 до 23) и количество минут (от 0 до 59).

Учтите, что число n может быть больше, чем количество минут в сутках.

```
N = int(input())
a = N//60%24
```

```
b = N%60
print(a,b)
```