ПРОГРАММИРОВАНИЕ НА PYTHON

ДОМАШНЕЕ ЗАДАНИЕ №1

ТЕМА: ПЕРЕМЕННЫЕ, ТИПЫ ДАННЫХ, ВВОД/ВЫВОД

ЗАДАНИЕ

Дорогие ребята!

В домашних заданиях по этому курсу я буду предлагать вам для решения по несколько простых задач для закрепления материала. Программный код для каждой задачи имеет смысл писать в отдельном файле, запуская этот файл из командной строки с помощью интерпретатора *python*. Всегда проверяйте работу своего кода!

Несмотря на то, что задачи действительно не сложные – некоторые из них у вас не получится написать с первого раза. Это совершенно нормально! Даже профессиональные программисты ошибаются. Главное – пробовать снова и снова! И учиться самостоятельно находить ошибки в коде. С некоторыми из них вам поможет сам интерпретатор, сопроводив сообщение об ошибке номером строки кода, на которой случилась ошибка, и поясняющим текстом. Если вы пока ещё не очень хорошо владеете английским, то копируйте текст ошибки в переводчик и старайтесь понять, в чём же дело.

Любой алгоритм в программировании можно реализовать множеством разных способов, всего лишь чуть иначе написав код. Не бойтесь проводить эксперименты с вашим кодом – это очень интересно!

Самым пытливым предлагаю задачи посложнее – они будут отмечены знаком змеи. Их невыполнение никак не штрафуется, но если кому-то удастся самостоятельно решить такую задачу, то на ближайшем занятии он получит максимальный балл и награду!

Для каждой задачи есть примеры ввода и вывода.

Ввод – это то, что пользователь печатает в консоли во время выполнения программы. Вывод – это тот результат, который ваша программа должна напечатать в консоли.

Старайтесь писать программу так, чтобы она выводила результат с таким же количеством и расположением пробелов и знаков препинания, как показано в примере вывода.

В разделе ПОДСКАЗКИ вы можете найти различные интересные способы использования синтаксиса языка, операторов и функций, которые мы ещё не успели рассмотреть на уроках.

Кроме того, не забывайте, что на странице в *MyStat* у вас есть раздел ДОПОЛНИТЕЛЬНЫЕ МАТЕРИАЛЫ, куда каждую неделю я буду добавлять для вас карточки со справочным материалом по пройдённым темам.

Код для решения каждой задачи необходимо сохранять в отдельный файл с расширением **.py** – так к концу курса у вас будут примеры программ, которые раскрывают все основные аспекты языка Python.

Успехов!

ЗАДАЧА 1

Написать программу для сбора данных о пользователе.

В первой строке попросить пользователя ввести имя.
Во второй строке попросить пользователя ввести фамилию.
В третьей строке попросить пользователя ввести год рождения.

Программа должна вывести сначала фамилию, потом имя, затем возраст пользователя. Возраст – число лет – считать без учёта дня и месяца рождения.

Для вывода используйте один вызов функции `print`.

Пример ввода:

```
Иван
Петров
2009
```

Пример вывода:

```
Петров Иван, 12
```

ЗАДАЧА 2

Напишите программу, которая считывает целое число, после чего на экран выводится следующее и предыдущее целое число с пояснительным текстом.

Пример ввода:

```
20
```

Пример вывода:

```
Следующее за числом 20 число: 21
Для числа 20 предыдущее число: 19
```

ЗАДАЧА 3

Напишите программу для пересчёта величины временного интервала, заданного в минутах, в величину, выраженную в часах и минутах.

Пример ввода:

```
150
```

Пример вывода:

```
150 мин - это 2 час 30 мин
```

ЗАДАЧА 4

Напишите программу, в которой рассчитывается сумма и произведение цифр положительного трёхзначного числа.

Пример ввода:

```
333
```

Пример вывода:

```
Сумма цифр = 9
Произведение цифр = 27
```

Сохраните код к каждой задаче в отдельный файл с расширением **.py**.
Упакуйте эти файлы в архив и прикрепите его в MyStat в качестве ответа.

ПОДСКАЗКИ

1. Интерпретатор python может работать в режиме консоли – выполняя команды по одной. Либо выполнять целиком файл с кодом.

   Чтобы перейти в режим консоли, необходимо в командной строке Windows ввести команду `python` и нажать Enter – после чего появится заголовок консольного режима интерпретатора и приглашение для ввода `>>>`
   Если это не происходит, то обратитесь повторно к Инструкции по подготовке.
   Чтобы выйти из консольного режима интерпретатора, необходимо вызвать функцию `quit()`, либо нажать Ctrl + Z и далее Enter.

2. Каждая задача предполагает написание отдельной программы, которую мы сохраняем в отдельный файл. Такие маленькие, но самостоятельные программы называют скриптами (англ. *script*) – я тоже буду их так называть.

   Итак, в Notepad++ набираем код скрипта, затем сохраняем его в папку с файлами по данному модулю.
   В командной строке Windows переходим в эту папку. Сделать это можно с помощью команды `cd /d “<путь к папке>”`. Данная команда также позволяет перейти в папку, расположенную на другом диске, например:

   ```
   C:\Users\Name>cd /d “D:\Documents\Python-Junior”

   D:\Documents\Python-Junior>
   ```

   В этих примерах серым цветом я обозначил печатаемое самой командной строкой *приглашение для ввода*, которое по умолчанию содержит текущий путь командной строки и символ `>`.
   (вид по умолчанию можно изменить с помощью системной переменной PROMPT)

   Теперь командная строка находится в той же папке, где расположен файл с нашим скриптом. А значит, мы можем запустить и протестировать его.
   Делается это с помощью команды `python`, за которой через пробел следует полное (с расширением) имя файла скрипта:

   ```
   D:\Documents\Python-Junior>python first.py
   Привет, мир!

   D:\Documents\Python-Junior>
   ```

Если имя файла содержит пробел, то после команды `python` имя файла необходимо заключить в двойные кавычки:

```
D:\Documents\Python-Junior>python “hello world.py”
Hello, world!

D:\Documents\Python-Junior>
```

Сразу после нашей команды выводится либо результат работы скрипта, либо сообщение об ошибке:

```
D:\Documents\Python-Junior>python test.py
  File "D:\Documents\Python-Junior\test.py", line 2
    name = input(,)
                 ^
SyntaxError: invalid syntax
```

Красным в тексте ошибки я выделил важные места: на какой строке и в каком месте строки (символ ^ указывает на это место) интерпретатор обнаружил ошибку, а также какую именно. Так, *invalid syntax* означает *недопустимый синтаксис* – то есть мы написали то, что не соответствует правилам языка Python.

3. При объявлении нескольких переменных, можно это делать в одну строку, перечисляя с одной стороны оператора присваивания имена переменных, а с другой стороны в том же порядке перечисляя значения переменных:

```
>>> a, b, c = 10, 20, 30
>>> print(a, b, c)
10 20 30
```

4. Каждый вызов функции `input()` сохраняет любое количество текста до первого перевода строки, то есть нажатия Enter. Таким образом, если нам необходимо получить от пользователя несколько строк, то необходимо несколько раз вызвать функцию `input()`:

КОМАНДНАЯ СТРОКА – CMD

```
...> python test.py
Ввод раз: строки
Ввод два: слипаются
строкислипаются
```

ФАЙЛ С КОДОМ – TEST.PY

```
s1 = input('Ввод раз: ')
s2 = input('Ввод два: ')
print(s1 + s2)
```

5. Вызовы функций могут вкладываться друг в друга:

```
>>> type(int(input()))
14
<class 'int'>
```

6. У функции `print` есть скрытый аргумент, который можно изменить. Он называется `sep` и отвечает за разделитель, который функция вставляет при выводе между несколькими аргументами – по умолчанию это пробел.

```
>>> print(1, 2)
1 2
>>> print(1, 2, sep='!')
1!2
>>> print(1, 2, sep='\n')
1
2
```

`'\n'` – это специальный символ переноса строки.