Лабораторная работа № 1: Ввод, вывод и типы данных

1. Запустите онлайн среду разработки "Tutorials Point" с рабочим пространством для языка программирования Python версии 3 по ссылке:

`https://www.tutorialspoint.com/execute_python3_online.php`

Интерфейс среды разработки состоит из трех областей: с левой стороны располагается каталог текущих файлов проекта, с правой стороны – редактор кода программы, под редактором кода находится терминал для ввода команд. В ходе выполнения лабораторных работ будет необходимо создавать программы и запускать их командами из терминала. Для сохранения файлов проекта можно использовать локальный диск ("Project" → "Download project") или диск Google ("Project" → "Save project" → "Save on Google Drive").

2. Создайте программу `hello-world.py` по следующему шаблону:

```
print("Hello World!");
```

Функция `print()` служит для вывода информации в терминал (консоль, командную строку).

3. Запустите программу, введя в терминале следующую команду:

```
python hello-world.py
```

В терминале будет выведено сообщение "Hello World!".

Запуск программ

Программы, написанные на языке программирования Python, запускаются с помощью команды `python <имя_файла>`. Python является некомпилируемым языком программирования, однако существуют специальные расширения для его компиляции в код на языке программирования Си. Следует отметить, что отсутствие точки с запятой в конце команды не является ошибкой для данного языка программирования.

4. Создайте новую программу `lab_01_01.py` по следующему шаблону:

```
# однострочный комментарий
'''
Многострочный
комментарий
'''

'''
    Пример работы с целыми числами
'''
a = 15
b = 7
print("a = ", a)
print("b = ", b)
print("a + b = ", a + b) # сложение чисел
```

```
print("a - b = ", a - b) # вычитание чисел
print("a * b = ", a * b) # умножение чисел
print("a / b = ", a / b) # деление чисел
print("a // b = ", a // b) # деление с округлением вниз до
целого
print("a % b = ", a % b) # остаток от деления
print("a ** b = ", a ** b) # возведение в степень
# приведение типа int к float
a1 = float(a)
print("a1 = ",a1)
print("\n\n")

'''
    Пример работы с вещественными числами
'''
c = 3.5
d = 6.9
print("c = ", c)
print("d = ", d)
print("c + d = ", c + d) # сложение чисел
print("c * d = ", c * d) # умножение чисел
print("c / d = ", c / d) # деление чисел

d1 = 2.3e-5
print("d1 = ",d1)
print("c + d1 = ", c + d1) # сложение чисел
# приведение типа float к int
d2 = int(d)
print("d2 = ",d2)
print("\n\n")

'''
    Операторы присваивания
'''
e = 7
print("e = 7: ", e)
e += 3
print("e += 3: ", e)
e -= 3
print("e -= 3: ", e)
e *= 15
print("e *= 5: ", e)
e /= 4
print("e /= 4: ", e)
e //= 2
print("e //= 2: ", e)
e %= 5
print("e %= 5: ", e)
e **= 3
print("e **= 3: ", e)
print("\n\n")
```

Удостоверьтесь в работоспособности программы, запустив ее через терминал. Ознакомьтесь с выведенной информацией. Результат выполнения программы приведен ниже:

```
a =  15
b =  7
a + b =  22
a - b =  8
a * b =  105
a / b =  2.142857142857143
a // b =  2
a % b =  1
a ** b =  170859375
a1 =  15.0



c =  3.5
d =  6.9
c + d =  10.4
c * d =  24.150000000000002
c / d =  0.5072463768115941
d1 =  2.3e-05
c + d1 =  3.500023
d2 =  6



e = 7:  7
e += 3:  10
e -= 3:  7
e *= 5:  105
e /= 4:  26.25
e //= 2:  13.0
e %= 5:  3.0
e **= 3:  27.0
```

Типы данных

В языке программирования Python используется динамическое определение типа данных. К простым типам данных относятся целые и вещественные числа, для которых доступны следующие арифметические операции:

- + (сложение)
- – (вычитание)
- * (умножение)
- / (деление)
- // (деление с округлением вниз до целого)
- % (остаток от деления)
- ** (возведение в степень)

Для присваивания переменной определенного значения используется оператор

присваивания: =. Также могут быть использованы сокращенные операции для автоматического присваивания переменной результата выполненного над ней выражения:

- `+=` (сложение с присваиванием)
- `-=` (вычитание с присваиванием)
- `*=` (умножение с присваиванием)
- `/=` (деление с присваиванием)
- `//=` (деление с округлением вниз до целого с присваиванием)
- `%=` (остаток от деления с присваиванием)
- `**=` (возведение в степень с присваиванием)

При использовании сокращенных операций результат выполнения `a += b` будет идентичен результату выполнения выражения `a = a + b`.

Для вещественных чисел язык Python поддерживает запись чисел в экспоненциальном формате, например, `0.5e4`, что эквивалентно $0.5*10^4$.

При необходимости привести целый тип к вещественному или же строковые данные к целому числу используется приведение типов. Для этого используются функции `int()`, `float()` и т.д., с указанием числа, которое необходимо привести к нужному формату в качестве аргумента функции в скобках.

5. Модифицируйте программу `lab_01_01.py`, изменив значения переменных `a` и `b`. Ознакомьтесь с результатом.

6. Модифицируйте программу `lab_01_01.py`, создав переменную `z` со значением 3 и переменную `y` со значением 5. Осуществите операции сложения, вычитания, умножения, деления, деления с округлением вниз, вычисления остатка от деления и возведения в степень над созданными переменными, выводя результаты на экран.

7. Дополните код программы `lab_01_01.py`. Создайте переменную `x` со значением 105 и `v` со значением 58. Осуществите вывод результата деления переменной `x` на переменную `v` на экран с приведением типа к вещественному и без приведения типов. Ознакомьтесь с результатом.

Логические и побитовые операции

В языке программирования Python дополнительно к основным существует булевский тип данных, для которого доступны два значения:

- `True`
- `False`

Для переменных и результатов выражений булевского типа доступны следующие логические операции:

- `not a` – логическое НЕ
- `a and b` – логическое И
- `a or b` – логическое ИЛИ
- `a == b` – эквиваленция
- `a != b` – проверка неравенства двух значений

- >, >=, <, <=

Python поддерживает возможность записи двусторонних условий, например:
`0 <= h <= 10`

Для работы с двоичными представлениями чисел предусмотрены побитовые операции:

- `x & y` – побитовое И
- `x | y` – побитовое ИЛИ
- `x ^ y` – побитовое исключающее ИЛИ
- `~x` – побитовая инверсия числа `x`
- `x >> n` – побитовый сдвиг числа `x` вправо на `n` бит
- `x << n` – побитовый сдвиг числа `x` влево на `n` бит

Для двоичного представления числового значения используется функция `bin()`, при этом число передается в качестве аргумента функции, например `bin(5)`.

8. Создайте новую программу `lab_01_02.py` по следующему шаблону:

```
'''
    Логические операции
'''
f = True
g = False
print("f: ", f)
print("not f: ", not f)
print("f and g: ", f and g)
print("f or g: ", f or g)
print("f == g: ", f == g)
print("f != g: ", f != g)
print("\n")
h = 3
i = 5
print("h = ", h)
print("i = ", i)
print("h > i: ", h > i)
print("h < i: ", h < i)
print("h >= i: ", h >= i)
print("0 < h <= i: ", 0 < h <= i)
print("\n\n")

'''
    Побитовые операции
'''
j = 7
k = 20
print("j = %d; j in binary format: %s" % (j, bin(j)) )
print("k = %d; k in binary format: %s" % (k, bin(k)) )
print("j & k: %d; binary: %s" % (j & k, bin(j & k)) ) #
побитовое AND
```

```
print("j | k: %d; binary: %s" % (j | k, bin(j | k)) ) #
побитовое OR
print("j ^ k: %d; binary: %s" % (j ^ k, bin(j ^ k)) ) #
побитовое XOR
print("~k: %d; binary: %s" % (~k, bin(~k)) ) # инверсия
двоичного числа
print("k>>1: %d; binary: %s" % (k>>1, bin(k>>1)) ) # сдвиг на
один бит вправо
print("k<<1: %d; binary: %s" % (k<<1, bin(k<<1)) ) # сдвиг на
один бит влево
print("\n\n")
```

Удостоверьтесь в работоспособности программы, запустив ее через терминал. Ознакомьтесь с выведенной информацией. Результат выполнения программы приведен ниже:

```
f:  True
not f:  False
f and g:  False
f or g:  True
f == g:  False
f != g:  True


h =  3
i =  5
h > i:  False
h < i:  True
h >= i:  False
0 < h <= i:  True



j = 7; j in binary format: 0b111
k = 20; k in binary format: 0b10100
j & k: 4; binary: 0b100
j | k: 23; binary: 0b10111
j ^ k: 19; binary: 0b10011
~k: -21; binary: -0b10101
k>>1: 10; binary: 0b1010
k<<1: 40; binary: 0b101000
```

9. Дополните код программы `lab_01_02.py`. Создайте переменные `A` и `B` с неравными целочисленными значениями, переменные `C` и `D` со значениями `True` и `False` соответственно.

10. Дополните код программы `lab_01_02.py`. Осуществите вывод на экран значений следующих логических выражений:

- ¬(C∧D)

- C∧D∨¬(C∧D)
- ¬C∨D

11. Дополните код программы lab_01_02.py. Осуществите вывод в консоль значений следующих выражений:
    - A<=B
    - A>B ∨ A==B
    - ¬(A<B)

12. Дополните код программы lab_01_02.py. Создайте переменную s со значением 154 и переменную p со значением 6. Осуществите вывод ее значения в десятичном и двоичном виде на экран.

13. Дополните код программы lab_01_02.py. Выполните операцию побитового ИЛИ над переменными s и p с записью результата в переменную s. Осуществите вывод значения в десятичном и двоичном виде на экран.

14. Дополните код программы lab_01_02.py. Выполните операцию побитового сдвига вправо на 2 бита над переменными s и p с записью результатов в соответствующие переменные. Осуществите вывод значений в десятичном и двоичном виде на экран.

15. Создайте новую программу lab_01_03.py по следующему шаблону:

```
'''
    Форматированный ввод/вывод данных
'''
m = 10
pi = 3.1415927
print("m = ",m)
print("m = %d" % m)
print("%7d" % m)
print("pi = ", pi)
print("%.3f" % pi)
print("%10.4f\n" % pi)
print("m = {}, pi = {}".format(m,pi))
ch = 'A'
print("ch = %c" % ch)
s = "Hello"
print("s = %s" % s)
print("\n\n")

code = input("Enter your position number in group: ")
n1, n2 = input("Enter two numbers splitted by space: ").split()
d, m, y = input("Enter three numbers splitted by \'.\':
").split('.')
print("{} + {} = {}".format(n1,n2,float(n1)+float(n2)))
print("Your birthday is %s.%s.%s and you are %d in the group
list" % (d,m,y,int(code)) )
```

Удостоверьтесь в работоспособности программы, запустив ее через терминал. Ознакомьтесь с выведенной информацией. Результат выполнения программы приведен ниже:

```
m =  10
m = 10
     10
pi =  3.1415927
3.142
    3.1416

m = 10, pi = 3.1415927
ch = A
s = Hello

Введите Ваш номер по списку группы:  10
Введите два числа через пробел:  4 7
Введите дату Вашего рождения:  01.01.1234
4 + 7 = 11.0
Ваш день рождения 01.01.1234 и Вы 10 в списке учебной группы
```

Форматированный вывод

Функция `print()` служит для вывода информации на экран. В языке программирования Python существует неформатированный вывод значений на экран, когда тип выводимого на экран значения определяется автоматически. В этом случае используется следующий вариант записи команды:

```
print("m = ", m)
```

При необходимости форматированного вывода значений в терминал используются различные спецификаторы для разных типов данных, в частности:

- `%i` или `%d` для целых чисел
- `%ld` для длинных целых чисел
- `%f` для вещественных чисел
- `%lf` для длинных вещественных чисел
- `%c` для символьного типа данных
- `%s` для строковых данных

В этом случае используется следующий вариант записи команды:

```
print("m = %d" % m)
```

Знак `%` перед именем переменной в аргументе функции предшествует перечисляемым аргументам.

Функция `print()` по умолчанию осуществляет вывод на новой строке. Для дополнительного перевода строки при выводе информации следует использовать `'\n'`. Символ `'\'` используется также для вывода на экран некоторых специальных символов, например, `'\'`, `'''`, `'"'`.

При выводе целочисленных и вещественных числовых значений возможно изменение базового спецификатора с целью округления выводимого числа:

- `%d` – вывод целого числа без форматирования
- `%3d` – вывод целого числа шириной как минимум 3 знака
- `%f` – вывод вещественного числа без форматирования
- `%4f` – вывод вещественного числа шириной как минимум 4 знака
- `%.5f` – вывод вещественного числа с точностью до 5 знака после запятой
- `%3.2f` – вывод вещественного числа шириной как минимум 3 знака с точностью до 2 знака после запятой

Пример наиболее часто используемых форматов округления:

Команда	**Результат**
`print("%d",305)`	`305`
`print("%4d",17)`	`  17`
`print("%f",7.2)`	`7.2`
`print("%.1f",10.87)`	`10.9`
`print("%3.2f",3.14159)`	`3.14`

При выводе нескольких аргументов может быть использован форматированный вывод, тогда все необходимые для вывода аргументы должны быть указаны в скобках после знака `%` в функции `print()`:

```
print("m = %d, n = %f" % (m,n))
```

Если тип выводимого значения точно не известен или же форматирование не требуется возможен вывод без указания формата (места вставки значений обозначаются пустыми фигурными скобками `{}`). В этом случае тип значения будет определен автоматически. Для этого используется функция `format()`, в качестве аргументов которой передаются значения для подстановки в выводимую строку. Функция `format()` вызывается от главного аргумента функции `print()`:

```
print("m = {}, n = {}".format(m,n))
```

Общий вид команды вывода в Python представляется как:

```
print(*objs, sep=' ', end='\n', file=sys.stdout, flush=False),
```

где:

- `*objs` – список выводимых значений;
- `sep` – разделитель, используемый при выводе, в частности для осуществления вывода значений в одну строку;
- `end` – символ, печатаемый по окончании вывода значений;
- `file` – поток вывода данных (консоль / файл);
- `flush` – автоматический выброс данных в поток.

Форматированный ввод

Ввод информации из терминала в языке программирования Python осуществляется с использованием функции `input()`. Для записи введенного значения в переменную необходимо присвоить возвращаемого функцией значение переменной, например:

```
a = input("Введите число: ")
```

В этом случае введенное число будет записано в переменную `a`.

При необходимости ввода нескольких значений с клавиатуры и их записи в различные переменные используется функция `split()`, вызываемая от возвращаемого функцией значения. Отсутствие аргумента функции `split()` означает, что разделитель для вводимых значений будет установлен как пробел (по умолчанию). В случае использования другого разделителя – необходимо указать его в скобках. Все значения будут считаны в строковом формате, для перевода их к числовому виду, необходимо выполнять приведение типов. Переменные, использующиеся для записи значений, в этом случае указывается перед оператором присваивания через запятую. Формат команды ввода в этом случае будет следующим:

```
a,b = input("Введите число: ").split()
```

или, в случае использования разделителя `'/'`

```
a,b = input("Введите два числа через \'/\': ").split('/')
```

16. Дополните код программы `lab_01_03.py`. Осуществите форматированный вывод в терминал значений переменных `m` и `pi`, подставив вместо многоточий их значения, в формате: "`m = ..; pi = ..`". При этом значение `m` должно быть выведено шириной 4 знака, а значение `pi` с точностью до 3 знаков после запятой.

17. Дополните код программы `lab_01_03.py`. Осуществите вывод в терминал значений переменных `m` и `pi`, подставив вместо многоточий их значения, в формате: "`m = ..; pi = ..`", используя автоматическое определение типов переменных с помощью функции `format()`.

18. Дополните код программы `lab_01_03.py`. Осуществите ввод значения номера Вашего курса обучения с клавиатуры, записав введенное значение в переменную `year`. Выведите значение переменной в терминал.

19. Дополните код программы `lab_01_03.py`. Осуществите ввод значений Ваших баллов ЕГЭ по русскому языку, математике и профильному предмету, записав введенные значения в переменные `r1`, `m1`, `p1`. В качестве разделителя при вводе значений используйте запятую. Выведите значения переменных в терминал.

20. Дополните код программы `lab_01_03.py`. Осуществите преобразование в десятичную систему счисления введенных двенадцатиразрядных чисел в системе счисления с основанием, которое равно увеличенному на два остатку от деления Вашего дня рождения на 8, и выведите результат преобразования в терминал.

21. Дополните код программы `lab_01_03.py`. Осуществите умножение и деление введенного с клавиатуры числа на два с использованием операции побитого сдвига влево и вправо соответственно. Выведите результат в терминал.