**Переменные**

Компьютер был создан для выполнения вычислений. Мы можем числа складывать, вычитать, умножать, делить и т.д.

Чтобы сохранить данные в программе используются **переменные**. Переменная это такая коробочка, куда мы что-то положим. Вы можете положить в переменную число 2. Для этого вы должны написать a = 2

Переменные в Python не нужно объявлять и не нужно указывать их тип. Переменная появляется в момент в момент присвоения ей значения.

Переменная имеет имя.

Переменные обозначаются латинскими буквами (заглавными или строчными), цифрами и знаком подчеркивания. Имя переменной должно начинаться не с цифры. Этот язык **регистрозависимый**. Существует разница между большими и маленькими буквами.

Имена переменных могут состоять из нескольких символов. В сложных программах переменные нужно называть так, чтобы было видно зачем нужна та или иная переменная.

Переменные можно называть русскими буквами, но так делать не принято.

Чтобы записать в переменную какое-нибудь значение нужно использовать **операцию присвоения**

а = 2

b = 3

c = a + b

Python вычислит сумму 2+3, получит 5 и запишет в переменную c.

**Операция присвоения** = записывает в переменную новое значение

**Арифметические операции**

+ - \*

/ (вещественное деление)

// (целочисленное деление)

% (остаток от деления)

\*\* возведение в степень

Операция возведения в степень выполняется справа налево

x \*\* 2 \*\* 3

x \*\* (2 \*\*3)

В противном случае нужны скобки

(x \*\* 2) \*\* 3

2 \*\* 1000 Python абсолютно точно посчитает значение выражения.

Когда вы выполняете арифметические операции и результатом является длинное целое число, т.е. число большее 8 байт, то Python переходит в длинную арифметику и будет работать со сколь угодно длинными числами, пока они помещаются в память вашего компьютера. Но с длинными целыми числами Python работает медленно. Можно не сильно задумываться о проблеме выбора типа для целочисленной переменной и проблеме переполнения.

**Переменные появляются в момент присвоения им значения.**

При этом, если в программе встретится команда

x = x +1

Но при этом ранее в переменную x ничего не было присвоено, то возникнет ошибка.

А вот, если будет написано

x = 4

x = x + 1

то значение переменной x станет равно 5. Будет взято значение 4, которое было присвоено в x первоначально, а затем оно будет увеличено на 1 и результат 5 будет вновь записан в переменную x/

**Сокращенная версия операций**

|  |  |
| --- | --- |
| a = a + 1 | a += 1 |
| a = a - 1 | a -= 1 |
| a = a \* 10 | a \*= 10 |
| a = a / 10 | a /= 10 |
| a = a // 10 | a //= 10 |
| a = a % 10 | a %= 10 |
| a = 0b = 0c = 0 | a = b = c = 0 |
| a = xb = xc = x | a = b = c = x |

**Типы данных**

Данные, с которыми вы работаете, могут быть разных типов.

**int** целочисленный тип a = 2

**float** вещественный (действительный) тип a = 2.7

Дробная часть отделяется точкой/

**str** строка a = "hello" или a = 'hello'. Строки заключаются в кавычки или апострофы. Это сделано для того, чтобы можно было выводить кавычки или апострофы внутри строки, например "it's". Но можно апостроф внутри строки задать так: 'it\'s' поставив перед ним ‘\’

**bool** логический тип a= **True** или a = **False**

Переменная может быть абсолютно любого типа. Вы можете вначале записать в нее число, а потом строку. Тип переменной определяется в тот момент, когда вы записываете туда новое значение.

Вы можете в любой момент поменять тип переменной, присвоив ей новое значение. Так делать не рекомендуется. Если вы собираетесь хранить разные по сути вещи, то лучше завести разные переменные. Особого смысла хранить в одной и той же переменной сначала строки, а потом числа не бывает.

С разными типами данных можно выполнять разные операции.

Операция сложения для чисел и для строк это разные операции.

a = 2 + 3 # 5

s = "Hello" + " World" # Hello World конкатенация строк

t= "2" + "3" #”23”

А вот числа со строками складывать нельзя

a = "2" + 3 будет ошибка

**Операции преобразования типов**

Если вы хотите получить конкатенацию строки и числа, то сначала нужно из числа получить строку.

Можно написать s="Hello " + "5"

Но если 5 хранится в переменной, то уже не получится

s = "Hello "

i = 5

t = s + i # ошибка

t = s + 'i' # Hello i

Нужно

t = s + str(i) # Hello 5

Из строки получить число можно так

s = "5"

d = int(s)

**Вывод данных**

Вывод значения переменной *a*

print(a)

Вывод значения переменных *a* и *b* через пробел

print(a, b)

Вывод двух слов через пробел

print("Hello", "world")

В строках могут использоваться специальные символы

\\ один символ \

\’ апостроф

\” кавычки

\n переход на новую строку

\t табуляция

Ниже два слова будут выведены через табуляцию

print("Hello \t world")

Вывод значения переменных *a* и *b* каждую с новой строки

print(a)

print(b)

**Параметры**

**sep** – разделитель

**end** – что выводится в конце строки

Вывод значения двух переменных без пробела

print(a, b, sep="")

Вывод двух переменных без перехода на новую cтроку (без пробела)

print(a, end="")

print(b)

Вывод значения переменных *a* и *b* каждую с новой строки

print(a,b,sep="\n")

Вывод арифметического выражения 1+2+3=6

print(1, 2, 3, sep="+", end="=")

print(6)

Вывод знака “\”

print(«\\»)

**Ввод данных**

**Считывание строки**

s = input()

**Считывание целого числа**

a = int(input())

Если передать функции **int** в качестве аргумента дробное число, то она сделает из него целое путём отбрасывания дробной части.

**Считывание вещественного числа**

a = float(input())

**Считывание двух чисел через пробел**

a, b = map(int, input().split())

**Задача.** Напишите программу, которая считывает с клавиатуры одно за другим два целых числа и выводит их сумму.

***Формат входных данных***

Два целых числа. Каждое с новой строки.

***Формат выходных данных***

Одно целое число.

Формат входных и выходных данных нужен, чтобы понимать данные какого типа будут подаваться на вход программы, и как нужно выводить ответ.

a = int(input())

b = int(input())

c = a + b

print(c)

Если бы данные вводились два числа через пробел, то программа имела бы вид

a, b = map(int, input())

c = a + b

print(c)

Можно не вводить переменную c, а сразу написать print(с).

При автоматической проверки задач в тестирующей системе не нужно выводить ни каких лишних пояснений.

Программа вида

print('Введите первое число')

a = int(input())

print('Введите второе число')

b = int(input())

c = a + b

print('Ответ')

print(c)

не будет засчитана. Будет выведена ошибка PE – ошибка представления данных. Т.к. все выводимые строки попадут в вывод программы. Проверяющая программа ждет в выводе Вашей программы одно целое число, а встречает три строчки и число. Это неверно.

**Ошибки**

**WA** – неверный ответ

**PE –**  ошибка представления данных

**TL** – превышен лимит времени. Программа работает слишком долго.

**ML** – превышен лимит памяти

**CE –** ошибка компиляции. В программе присутствуют синтаксические ошибки

**Математические функции**

**Модуль числа** abs()

x = abs(-34)

Большинство математических функций находится в модуле **math**, который нужно импортировать в начале программы.

Квадратный корень **math.sqrt**(x), впрочем, его можно найти, как x \*\* 0.5.

Число π **math.pi**

Например, чтобы зная радиус найти площадь круга, можно написать такую программу

**import** math

r = **float**(input())

s = math.pi \* r \*\* 2

print(s)

Если мы хотим импортировать не весь модуль, а только конкретную функцию, то это можно записать так. При этом при вызове функции не нужно указывать имя модуля.

**from** math **import** pi

r = **float**(input())

s = pi \* r \*\* 2

print(s)