|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Арифметические операторы в Python**  Арифметические операторы используются с числовыми значениями для выполнения общих математических операций: | | | | **Операторы присваивания в Python**  Операторы присваивания используются для присваивания значений переменным: | |
| ***Оператор*** | ***Значение*** | | | ***Оператор*** | ***Пример*** |
| \*\*  \*  /  %  //  +  — | возведение в степень  умножение  деление  остаток от деления  целочисленное деление  добавление (unary first)  вычитание (unary first) | | | =  +=  -=  \*=  /=  %=  //=  \*\*=  &=  |=  ^=  >>=  <<= | x = 5 x = 5  x += 3 x = x + 3  x -= 3 x = x — 3  x \*= 3 x = x \* 3  x /= 3 x = x / 3  x %= 3 x = x % 3  x //= 3 x = x // 3  x \*\*= 3 x = x \*\* 3  x &= 3 x = x & 3  x |= 3 x = x | 3  x ^= 3 x = x ^ 3  x >>= 3 x = x >> 3  x <<= 3 x = x << 3 |
| **Операторы сравнения в Python**  Операторы сравнения используются для сравнения двух значений: | | | | | |
| ***Оператор*** | | | ***Значение*** | | |
| >  <  >=  <=  ==  != | | | больше чем  меньше чем  больше чем или равно  меньше чем или равно  Равно  не равно | | |
| **Операторы тождественности в Python**  Операторы тождественности используются для сравнения объектов. Являются ли они одним и тем же объектом с одинаковым местоположением в памяти: | | | | | |
| ***Оператор*** | | | ***Значение*** | | |
| is  is not | | | Возвращает true если переменные являются одним объектом  Возвращает true если переменные разные | | |
| **Операторы принадлежности в Python**  Операторы принадлежности используются для проверки того, представлена ​​ли последовательность в объекте: | | | | | |
| ***Оператор*** | | ***Оператор*** | | | |
| in  not in | | Возвращает True если последовательность присутствует в объекте  Возвращает True если последовательность не присутствует в объекте | | | |
|  | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Логические операторы в Python**  Логические операторы используются для объединения условных операторов: | | | |
| ***Оператор*** | ***Значение*** | | ***Пример*** |
| and  or  not | Возвращает значение True если оба утверждения верны  Возвращает True если одно из утверждений верно  Меняет результат, возвращает False если результат True | | x < 5 and x < 10  x < 5 or x < 4  not(x < 5 and x < 10) |
| **Побитовые операторы в Python**  Побитовые операторы используются для работы в битовом (двоичном) формате: | | | |
| ***Оператор*** | ***Название*** | ***Значение*** | |
| &  |  ^  ~  <<  >> | И  Или  только или  Не  Сдвиг влево  Сдвиг вправо | Устанавливает каждый бит в 1, если оба бита 1  Устанавливает каждый бит в 1 если один из двух битов 1  Устанавливает каждый бит в 1 если только один из битов 1  Переставляет все биты  Сдвигает влево на количество бит указанных справа  Сдвигает вправо на количество бит указанных справа | |
| **Типы данных в Python** | | | |
| ***Оператор*** | ***Название*** | ***Значение*** | |
| type(...)  str()  int()  float()  bool() | тип  текст  целые цифр  дробь  true or false | определение типа значения  проверка соответствия значения | |
| **Операторы условий (loop) в Python** | | | |
| ***Оператор*** | ***Название*** | ***Значение*** | |
| If ... :  else :  elif ... :  while … :  for ... in range(a,b,c):  time.sleep(sec)  break  continue | Если  Иначе  иначе-если  Покуда вo  Для в пределе  Ожидание  Окончить  продолжить | действие будет выполняться если соответствует  действие будет выполняться если не соответствует  действие будет выполняться иначе, если соответствует  действие выполняется бесконечно, пока соответствует  выбор значения из списка (базы) (начинает с ‘a’ и до значения ‘b-1’ c отступом 'с')  задержка в выводе значений в периоде ‘sec’  оканчивает условие при достижении значения  выполняет все условия пропуская(кроме) заданного значения | |
| **Операторы действия в Python** | | | |
| ***Оператор*** | ***Название*** | ***Значение*** | |
| print(...)  input(…)  import …  exit()  print (f ”{verb} ... {verb} ...”)  try  except …Error  return ...  \n  # … | Импортировать  F соединение  Сделай  Исключение  Возвращать  Комментарий | вывести значение на экран  разрешает пользователю ввести значение  Загрузить (импортировать) модуль (библиотеку)  закончить действие  Соединяет разные типы значений  Выполнение кода без разрешения  Выполнение исключения вместо системной ошибки  Возврат значения из функции  Начать с новой строки (переход)  Строка с данным символом не запускается как команда | |
| **Методы список (list), библиотеки (dictionary) и картеж (tuple) в Python** | | | |
| ***Оператор*** | ***Название*** | ***Значение*** | |
| mth  (…, …)  {... : …, key : value}  [ …, … ]  *for* key, value*in* … .items():  .append(…) или .update({… : …})  .insert(… , …)  .pop(...)  del …[…]  .union(...)  .intersection(...)  ... .difference(...)  len(...)  min(…) / max(…)  … .sort()  … .reverse()  … .count(...)  … .index()  … .split()  replace("...","...")или ...[a] = b  … .copy() | Методы  Создать  Создать  Создать  Добавить  Добавить  Добавить  Удалить  Удалить  Объединить  Отличие  Кол-во  Сортировка  разворот  замена  копирование | Создание картеж (tuple) – последов. **неизменяемых** значений (меньше размер хранения)  Создание библиотеки (dictionary) - последов. **неизменяемых keys** значений  Создание списка (list)  Вывести все ключи и значения из библиотеки (перебрать или проверить всю библиотеку)  Добавить значение в конец списка (list)  Добавить значение в конец библиотеки (dictionary)  добавить значение в определенном месте в группе (list)  Удалить значение из библиотеки (dictionary)  Удалить последнее значение из списка (list)  Соединять списка (list), библиотеки, списки без повторения значений  Соединяет и отображает только повторяемые значения  Выводит значение из одного списка (list) кот нет в другом  вывести кол-во значений в списка (list)  как часто упоминается значение в список (list), библиотеки (dictionary) и картеж (tuple)  найти позицию в список (list), библиотеки (dictionary) и картеж (tuple)  разделяет предложение на отдельные слова  замена значения в списке где а – позиция(key), в – новое значение  создает копию или копирует библиотеку | |
| **Операторы функции в Python** | | | |
| ***Оператор*** | ***Название*** | ***Значение*** | |
| def ...():  print(...)  return …  ...()  Def ...(first. Last)  Print (...)  ...(first,last) |  | создание функции  вывести на экран  Возврат значения из функции, выполняет без вывода на экран  вызвать функцию  добавление нескольких переменных в функцию | |

def calculate(operator, x, y):

if operator == "+":

print(x + y)

elif operator == "-":

print(x - y)

else:

print(f"unknown: {operator}")

calculate("-", 30, 10)

results = [3.12, 8.2, 7]

corrected = [result + 1 for result in results]

print(corrected)

class Computer:

def \_\_init\_\_(self, size, storage):

self.size = size

self.storage = storage

def print\_specs(self):

print("Display size: " + self.size)

print("Storage size: " + self.storage)

low\_spec = Computer("13", "256GB")

high\_spec = Computer("27", "1TB")

print("Low Spec Computer:")

low\_spec.print\_specs()

print("High Spec Computer:")

high\_spec.print\_specs()

IL – instruction list

**COMPILATION** - the source program is translated once (however, this act must be repeated each time you modify the source code) by getting a file (e.g., an .exe file if the code is intended to be run under MS Windows) containing the machine code; now you can distribute the file worldwide; the program that performs this translation is called a compiler or translator;

**INTERPRETATION** - you (or any user of the code) can translate the source program each time it has to be run; the program performing this kind of transformation is called an interpreter, as it interprets the code every time it is intended to be executed; it also means that you cannot just distribute the source code as-is, because the end-user also needs the interpreter to execute it.

**variables** - the name itself suggests that the content of these containers can be varied in (almost) any way.

They are called **keywords** or (more precisely) **reserved keywords**. They are reserved because **you mustn't use them as names**:

['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

**FUNCTION:**

* if a particular fragment of the code begins to appear in more than one place, consider the possibility of isolating it in the form of a function
* if a piece of code becomes so large that reading and understating it may cause a problem, consider dividing it into separate, smaller problems, and implement each of them in the form of a separate function

**single integer value mustn't be iterated through by the**for**loop**.