|  |
| --- |
| function\_name(argument)  print("Hello, World!") |

|  |
| --- |
| The backslash (\)  **print("The itsy bitsy spider\nclimbed up the waterspout.")**  **print()**  **print("Down came the rain\nand washed the spider out.")** |

|  |
| --- |
| The arguments are **separated by commas**.  print("The itsy bitsy spider" , "climbed up" , "the waterspout.")  The itsy bitsy spider climbed up the waterspout. |

# keyword arguments (end)

* a keyword argument consists of three elements: a **keyword** identifying the argument (end here); an **equal sign** (=); and a **value** assigned to that argument;
* any keyword arguments have to be put **after the last positional argument** (this is very important)

|  |
| --- |
| print("My name is", "Python.", end=" ")  print("Monty Python.")  print("nata")  Consol:  My name is Python. Monty Python.  nata |

keyword arguments sep (like separator).

|  |
| --- |
| print("My", "name", "is", "Monty", "Python.", sep="-")  Consol:  My-name-is-Monty-Python. |

# Integers

# Therefore, you can write this number either like this: 11111111, or like that: 11\_111\_111.

# Arithmetic operators: exponentiation

|  |
| --- |
| **23**. Pure text editors don't accept that, so Python uses \*\* instead, e.g., 2 \*\* 3. |

# Arithmetic operators: division

|  |
| --- |
| print(6 / 3)  print(6 / 3.)  print(6. / 3) print(6. / 3.)The result is always a floatA // (double slash) sign is an **integer divisional** |

# Operators: remainder (modulo)

|  |
| --- |
| print(14 % 4)   * 14 // 4 gives 3 → this is the integer **quotient**; * 3 \* 4 gives 12 → as a result of **quotient and divisor multiplication**; * 14 - 12 gives 2 → this is the **remainder**.  Consol 2 |

# Correct and incorrect variable names

|  |
| --- |
| And now for some **incorrect names**:  10t (does not begin with a letter), Exchange Rate (contains a space)  ['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'] |

# Creating variables

|  |
| --- |
| var = 1  print(var)    It consists of two simple instructions:   * The first of them creates a variable named var, and assigns a literal with an integer value equal to 1.  The second prints the value of the newly created variable to the console. |

# Printing string and integer in one line

|  |
| --- |
| print("c =", c) |

# Shortcut operators

|  |
| --- |
| For example, if we need to calculate a series of successive values of powers of 2, we may use a piece like this:  x = x \* 2  You may use an expression like this if you can't fall asleep and you're trying to deal with it using some good, old-fashioned methods:  sheep = sheep + 1  Python offers you a shortened way of writing operations like these, which can be coded as follows:  x \*= 2  sheep += 1 |

To comment and uncomment in python

**CTRL** + **/**

# The input() function

|  |
| --- |
| print("Tell me anything...")  anything = input()  print("Hmm...", anything, "... Really?")  input()**function is a string**. |