Built-ins

ID	Name	Description	Module	Step	Explanation	Example
1	`print`	print	1	2	Prints the value to the screen.	`print(5)` prints the number 5 to the screen.
2	`+`	addition	1	3	Adds two numbers.	`2 + 3` has the value `5`.
3	`-`	subtraction	1	3	Subtracts the first number by the second.	`3 - 1` has the value `2`.
4	`*`	multiplication	1	3	Multiplies two numbers.	`4 * 3` has the value `12`.
5	`/`	division	1	3	Divides the first number by the second.	`12 / 3` has the value `4.0` since division always results in a floating point number.
6	`+`	unary plus	1	3	Used on one value, does not change the number.	`+ 3` has the value `3`.
7	`-`	negation	1	3	Used on one value, negates the number.	`- 3` has the value `-3`.
8	`//`	quotient	1	3	Calculates the quotient when the first number is divided by the second.	`20 % 7` has the value `2`, since the integer part of `20` divided by `7` is 2.
9	`%`	remainder	1	3	Calculates the remainder when the first number is divided by the second.	`20 // 7` has the value `6`, since the remainder when dividing `20` by `7` is `6`.

ID	Name	Description	Module	Step	Explanation	Example
10	`**`	exponentiation	1	3	Raises the first number to the power of the second number.	`2 ** 3` has the value `8`, since it is the cube of `2`.
11	`import`	import	2	3	Used to import a module.	`import math` can be used to import the math module.
12	`math`	math module	2	3	The math module. Contains functions and constants relating to math.	After the math module is imported, `math.sqrt` and other functions and constants can be used.
13	`sqrt`	square root	2	3	Determines the square root of the input. In the math module.	After the math module is imported, `math.sqrt(4)` has the value `2.0`, since taking the square root always results in a floating point number.
14	`pi`	pi	2	3	The constant (\pi). In the math module.	After the math module is imported, the constant can be accessed as `math.pi`.
15	`dir`	directory	2	3	Used with the input `__builtins__` (to list all built-ins), a value (to list all built-ins applicable to that value), or a module (to list all contents of the module).	Using `dir(9)` results in all functions that can be used on `9`.
16	`from math import`	import a single function or constant	2	3	Used to import a single function or constant from a module.	After using `from math import sqrt` one can use `sqrt` instead of `math.sqrt`.

ID	Name	Description	Module	Step	Explanation	Example
17	`pow`	exponentiation	2	3	Uses the first input as the base and the second as the exponent. There are two versions. The version in the math module always returns a floating point number.	`pow(2, 3)` has the value `8` and `math.pow(2, 3)` has the value `8.0`.
18	`abs`	absolute value	2	3	Produces the absolute value of the input.	`abs(-3.4)` has the value `3.4`.
19	`__doc__`	documentation	2	3	Prints the docstring. Uses dot notation.	The docstring for `abs` has the lines `abs(number) -> number` and `Return the absolute value of the argument.`.
20	`input`	user input	2	4	Prompts the user to type a value. The optional string is the prompt.	The function call `input("Enter your age:")` prompts the user for input with `"Enter your age:"`.
21	`type`	determine type	2	4	Produces the type of the input.	The value of `type(5)` is `<class 'int'>`, indicating that `5` is an integer.
22	`int`	make into integer	2	4	Makes an integer, floating point number, or string (if an integer in quotation marks) into an integer.	The values of `int(5.3)` and `int("5")` are both `5`.

ID	Name	Description	Module	Step	Explanation	Example
23	`float`	make into floating point	2	4	Makes an integer, floating point number, or string (if a number in quotation marks) into a floating point number.	The values of `float(5)` and `float("5")` are both `5.0`.
24	`str`	make into string	2	4	Makes a number or a string into a string.	The values of `str(5)` and `str(5.0)` are the strings `"5"`, and `"5.0"`, respectively.
25	`len`	length	2	7	Produces the length of a string.	The values of `length("")` and `length("cat")` are 0 and 3, respectively.
26	`[i]`	index	2	7	Produces the character in position `i` of a string.	The value of `"cat"[1]` is `"a"`, since the first position is at index zero.
27	`+`	concatenation	2	7	Produces the string formed by gluing together the input strings.	The value of `"hot" + "dog"` is the string `"hotdog"`.
28	`*`	repeated concatenation	2	7	Produces the string formed by repeatedly gluing the input with copies of the input.	The value of `2 * "no"` is `"nono"`.
29	`[a:b]`	slice	2	7	Produces a substring from positions `a` up to but not including position `b`.	The value of `"hotdog"[2:5]` is `"tdo"`.

ID	Name	Description	Module	Step	Explanation	Example
30	`upper`	upper case	2	8	Produces a string with all lower-case letters replaced by upper-case letters. Uses dot notation.	The value of `"Ha!".upper()` is `"HA!"`.
31	`lower`	lower case	2	8	Produces a string with all upper-case letters replaced by lower-case letters. Uses dot notation.	The value of `"Ha!".lower()` is `"ha!"`.
32	`ceil`	ceiling	2	8	Produces the integer reached by pushing up to the ceiling. In the math module.	The values of `math.ceil(2)`, `math.ceil(1.2)`, and `math.ceil(-1.2)` are `2`, `2`, and `-1`, respectively.
33	`floor`	floor	2	8	Produces the integer reached by pushing down to the floor. In the math module.	The values of `math.floor(2)`, `math.floor(1.2)`, and `math.floor(-1.2)` are `2`, `1`, and `-2`, respectively.
34	`trunc`	truncation	2	8	Produces the integer formed by cutting off all digits after the decimal point. In the math module.	The values of `math.trunc(2)`, `math.trunc(1.2)`, and `math.trunc(-1.2)` are `2`, `1`, and `-1`, respectively.
35	`True`	true	5	3	The Boolean constant True.	If the value of a Boolean is not `True`, it must be `False`.
36	`False`	false	5	3	The Boolean contant False.	If the value of a Boolean is not `False`, it must be `True`.

ID	Name	Description	Module	Step	Explanation	Example
37	`or`	or	5	3	Produces `True` when at least one smaller expression is True.	The value of `True or False` is `True`.
38	`and`	and	5	3	Produces `True` when both smaller expressions are True.	The value of `True and True` is `True`.
39	`not`	not	5	3	Produces `True` when the value is False and False when the value is True.	The value of `not True` is `False`.
40	<	less than	5	3	Produces `True` when the first input has a value less than the second input.	The value of `3 < 2` is `False`.
41	>	greater than	5	3	Produces `True` when the first input has a value greater than the second input.	The value of `5 > 2` is `True`.
42	<=	less than or equal to	5	3	Produces `True` when the first input has a value less than or equal to the second input.	The value of `3 <= 3` is `True`.
43	>=	greater than or equal to	5	3	Produces `True` when the first input has a value greater than or equal to the second input.	The value of `5 >= 10` is `False`.
44	==	equals	5	3	Produces `True` when the first input has a value equal to the second input.	The value of `1 == 0` is `False`.

ID	Name	Description	Module	Step	Explanation	Example
45	`!=`	not equals	5	3	Produces `True` when the first input has a value not equal to the second input.	The value of `2 != 2` is `True`.
46	`is`	is	5	3	Produces `True` when the first and second input are at the same memory address. Syntax is `a is b`.	For a variable `num`, the value of `num is num` is `True`.
47	`is not`	is not	5	3	Produces `True` when the first and second input are not at the same memory address. Syntax is `a is not b`.	The value of `a is not b` is `True` whenever `a is b` is `False`.
48	`ord`	code point from character	5	3	Produces the code point corresponding to the input character.	The value of `ord("A")` is `65`.
49	`chr`	character from code point	5	3	Produces the character corresponding to the input code point.	The value of `chr(65)` is `"A"`.
50	`isalpha`	letter check	5	3	Produces `True` if the input string is made up of letters. Uses dot notation.	The value of `"Cat".isalpha()` is `True`.
51	`islower`	lower case check	5	3	Produces `True` if the input string is made up of lower-case letters. Uses dot notation.	The value of `"cat".islower()` is `True`.

ID	Name	Description	Module	Step	Explanation	Example
52	`isupper`	upper case check	5	3	Produces `True` if the input string is made up of upper-case letters. Uses dot notation.	The value of `"CAT".isupper()` is `True`.
53	`isdigit`	digit check	5	3	Produces `True` if the input string is made up of digits. Uses dot notation.	The value of `"123".isdigit()` is `True`.
54	`isspace`	space check	5	3	Produces `True` if the input string is made up of blank spaces. Uses dot notation.	The value of `" ".isspace()` is `True`.
55	`assert`	assertion	7	7	Results in an assertion error if the Boolean expression after `assert` is not True.	Running a program containing the line `assert 1 == 0, "No way"` results in an assertion error with the message `"No way"`.
56	`len`	list length	9	2	Produces the length of a list.	The value of `len([1, 2, 3])` is 3, since the number of items in the list is 3.
57	`[i]`	index	9	2	Produces the item in position `i` of a list.	The value of `[1, 2, 3][1]` is 2, since 2 is the item in position 1 of the list.
58	`+`	concatenation	9	2	Produces the list formed by gluing together the input lists.	The value of `[0, 1] + [2, 3]` is the list `[0, 1, 2, 3]`.
59	`*`	repeated concatenation	9	2	Produces the list formed by repeatedly gluing the input with copies of the input.	The value of `[3, 4] * 2` is `[3, 4, 3, 4]`, as it is formed by concatenating two copies of `[3, 4]`.

ID	Name	Description	Module	Step	Explanation	Example
60	`[a:b]`	slice	9	2	Produces a list from positions `a` up to but not including position `b`.	The value of `[6, 7, 8, 9][1:3]` is `[7, 8]`, as it is formed of the items in positions 1 through 2.
61	`min`	minimum	9	3	Produces the minimum item in the list. The input list does not change.	The value of `min([4, 3, 6])` is `3`.
62	`max`	maximum	9	3	Produces the maximum item in the list. The input list does not change.	The value of `max([4, 3, 6])` is `6`.
63	`count`	count	9	3	Produces the number of times the input item appears in the input list. Uses dot notation. The input list does not change.	The value of `[8, 7, 8, 9].count(8)` is `2`, since the value `8` appears two times in the list.
64	`index`	search	9	3	Produces the smallest index of a position containing the input item in the input list. Uses dot notation. The input list does not change.	The value of `[8, 9, 7].index(9)` is `1`, since 9 is in position 1.
65	`in`	membership	9	3	Produces `True` if the item is in the list. Syntax is `a in b`.	The value of `9 in [8, 9, 7]` is `True`.
66	`list`	new list	9	3	Produces a new list containing the items of the input as items. The input list does not change.	The value of `list("ape")` is `["a", "p", "e"]`.

ID	Name	Description	Module	Step	Explanation	Example
67	`sorted`	sorted	9	3	Produces a new list with the input items in sorted order. The input list does not change.	The value of `sorted([6, 3, 5])` is `[3, 5, 6]`.
68	`append`	append	9	3	Mutates the input list by appending the input item to the end of the list.	For a variable `seq` with value `[1, 2]`, using `seq.append(3)` results in `seq` becoming the list `[1, 2, 3]`.
69	`remove`	remove	9	3	Mutates the input list by removing the first occurrence of the input item from the list.	For a variable `seq` with value `[7, 8, 7]`, using `seq.remove(7)` results in `seq` becoming the list `[8, 7]`.
70	`insert`	insert	9	3	Mutates the input list by inserting at the input index the input item.	For a variable `seq` with value `[7, 8, 7]`, using `seq.insert(2, 9)` results in `seq` becoming the list `[7, 8, 9, 7]`.
71	`sort`	sort	9	3	Mutates the input list by sorting the items.	For a variable `seq` with value `[4, 2, 3]`, using `seq.sort()` results in `seq` becoming the list `[2, 3, 4]`.

ID	Name	Description	Module	Step	Explanation	Example
72	`range`	range	10	2	Produces an immutable sequence of integers starting at `start` and stopping before `stop` at intervals of `step`. Input order is `(start, stop, step)`. If `step` is missing, its default value is 1. If both `step` and `stop` are missing, their default values are 1 and 0, respectively.	The value of `range[3, 9, 2]` is a structure with the values 3, 5, and 7.
73	`split`	split	10	8	Produces a list of strings from an input string. Uses dot notation.	The value of `"a b c".split()` is `['a', 'b', 'c']`.
74	`class`	class definition	11	2	Creates a new class.	To create a class `Oyster`, the first line should be `class Oyster:` and the second line, indented, should have the docstring.
75	`isinstance`	instance check	11	2	Produces `True` if the first input (an object) is a member of the second input (a class).	The value of `isinstance(eye, Circle)` is `True` if we have created an object `eye` of type `Circle`.
76	`hasattr`	attribute check	11	2	Produces `True` if the first input (an object) has an attribute with the second input (a string) as its name.	The value of `hasattr(eye, "colour")` is `True` if the object `eye` is of a type with attribute `colour`.

ID	Name	Description	Module	Step	Explanation	Example
77	`dir`	directory	11	2	Used with an class name as input to show contents of a class.	Using `dir(Circle)` shows contents of the class `Circle`.
78	`copy`	copy module	11	3	Contains functions related to copying.	To load the copy module, use `import copy`.
79	`copy`	copy function	11	3	Makes a shallow copy. In the copy module.	To create a copy of an object `eye`, use `copy.copy(eye)`. If the object contains other objects, those objects will not be copied.
80	`deepcopy`	deep copy function	11	3	Makes a deep copy. In the copy module.	To create a copy of an object `eye` in which all objects within it are also copied, use `copy.deepcopy(eye)`.
81	`len`	tuple length	12	3	Produces the length of a tuple.	The value of `len((1, 2, 3))` is 3, since the number of items in the tuple is 3.
82	`[i]`	index	12	3	Produces the item in position `i` of a tuple.	The value of `(1, 2, 3)[1]` is 2, since 2 is the item in position 1 of the tuple.
83	`+`	concatenation	12	3	Produces the tuple formed by gluing together the input tuples.	The value of `(0, 1) + (2, 3)` is the tuple `(0, 1, 2, 3)`.
84	`*`	repeated concatenation	12	3	Produces the tuple formed by repeatedly gluing the input with copies of the input.	The value of `(3, 4) * 2` is `(3, 4, 3, 4)`, as it is formed by concatenating two copies of `(3, 4)`.

ID	Name	Description	Module	Step	Explanation	Example
85	`[a:b]`	slice	12	3	Produces a tuple from positions `a` up to but not including position `b`.	The value of `(6, 7, 8, 9)[1:3]` is `(7, 8)`, formed of the items in positions 1 through 2.
86	`tuple`	tuple	12	3	Produces a tuple from the items in the input.	The value of `tuple([1, 2])` is `(1, 2)`.
87	`min`	minimum	12	3	Produces the minimum item in the tuple.	The value of `min((4, 3, 6))` is `3`.
88	`max`	maximum	12	3	Produces the maximum item in the tuple.	The value of `max((4, 3, 6))` is `6`.
89	`count`	count	12	3	Produces the number of times the input item appears in the input tuple. Uses dot notation.	The value of `(8, 7, 8, 9).count(8)` is `2`, since the value `8` appears two times in the tuple.
90	`index`	search	12	3	Produces the smallest index of a position containing the input item in the input tuple. Uses dot notation.	The value of `(8, 9, 7).index(9)` is `1`, since 9 is in position 1.
91	`in`	membership	12	3	Produces `True` if the item is in the tuple. Syntax is `a in b`.	The value of `9 in (8, 9, 7)` is `True`.
92	`list`	new list	12	3	Produces a new list containing the items of the input tupleas items.	The value of `list(("a", "p", "e"))` is `["a", "p", "e"]`.

ID	Name	Description	Module	Step	Explanation	Example
93	`sorted`	sorted	12	3	Produces a new list with the tuple input items in sorted order.	The value of `sorted((6, 3, 5))` is `[3, 5, 6]`.
94	`[i]`	access dictionary item	12	4	Produces the value associated with key `i`.	For a variable `data` assigned the value `{'one': 'un', 'two': 'deux'}`, the value of `data['two']` is `deux`.
95	`keys`	keys	12	4	Produces a list of keys in the dictionary. Uses dot notation.	For a variable `data` assigned the value `{'one': 'un', 'two': 'deux'}`, using `data.keys()` results in the keys `one` and `two`.
96	`items`	key, item pairs	12	4	Produces a list of key, item pairs. Uses dot notation.	For a variable `data` assigned the value `{'one': 'un', 'two': 'deux'}`, using `data.items()` results in the pairs `('one', 'un')` and `('two', 'deux')`.
97	`in`	in	12	4	Produces `True` if the first input is a key in the second input (a dictionary). Syntax is a in b.	For a variable `data` assigned the value `{'one': 'un', 'two': 'deux'}`, the value of `'two' in data` is `True`.
98	`not in`	not in	12	4	Produces `True` if the first input is not a key in the second input (a dictionary). Syntax is a not in b.	For a variable `data` assigned the value `{'one': 'un', 'two': 'deux'}`, the value of `'three' not in data` is `True`.
99	`zip`	zip	12	4	Produces pairs of values from two input sequences.	Using `zip('ha','ho')` results in the pairs `('h', 'h')` and `('a', 'o')`.