

Expression	Action	Example
print()	Display the result of a command	<pre>x="Hello world" print(x)  output: Hello world print(input("what is your name?"))</pre>
input()	Collect inputs from users	<pre>output: what is your name? x="Regular expressions" type(x)</pre>
type()	Find the type of a variable	<pre>output: &lt;class 'str'&gt; len([1, 2, 3])</pre>
len()	Find the number of items in a variable	<pre>output: 3 print("I want you to add\"")</pre>
\	Escape a character that changes the intent of a line of code	<pre>output: I want you to add"" print("This is a line \n This is a second line")</pre>
\n	Break a string character to start on the next line	<pre>output: This is a line This is a second line</pre>
def function_name(parameter): commands	Initiate a function with an optional parameter	<pre>def yourName(x): print(x+1) add_3_to = lambda y: y+3 print(add_3_to(4))</pre>
lambda	Call an anonymous function	<pre>output: 7 def yourName(x): return x+1 class myClass: def myFunc(x): class myClass: def __init__(self, attributes...) Rename a file containing a module as:</pre>
return	Return a result from a function	
class	Create a Python object	
def __init__	Initialize the attributes of a class	
"__init__.py"	Save a file containing a module so that it's read successfully in another Python file	<pre>"__init__.py" int(1.234)</pre>
int()	Convert a variable to integer	<pre>output: 1 str(1.234)</pre>
str()	Convert a variable to string	<pre>output: '1.234' float(23)</pre>
float()	Convert a variable to float	<pre>output: 23.0 from collections import Counter dict(Counter([1,1,2,1,2,3,3,4]))</pre>
dict(Counter())	Convert a list or a tuple into a dictionary after sorting with a Python built-in Counter	<pre>output: {1: 3, 2: 2, 3: 2, 4: 1} round(23.445)</pre>
round()	Round up the output of an operation to the nearest whole number	<pre>output: 23 round(23.4568, 2)</pre>
round(operation or number, decimal places)	Round up the output of an operation to a specific number of decimal places	<pre>output: 23.46 if 2&lt;3: print("Two is smaller") if 2&lt;3: print("Two is smaller") elif 2==3: print("Go on") if 2&lt;3: print("Two is smaller") elif 2==3: print("Go on") else: print("Three is greater")</pre>
if:	Initiate a conditional statement	
elif:	Make a counterstatement when the if statement is False	
else:	Make a final counterstatement if other conditions are False	

		<pre> a=[1, 4, -10, 6, 8] for b in a:     if b&lt;=0:         continue     print(b)  output: 1 4 6 8 </pre>
continue	Ignore a condition and execute the rest of the loop	
		<pre> a=[1, 4, -10, 6, 8] for b in a:     if b&gt;=6:         break     print(b)  output: 1 4 -10 </pre>
break	Terminate the flow of a loop with a given condition	
pass	Ignore a set of prior instructions	<pre> for b in a:     pass try:     print(a)  except:     print("An error occurred!")  output: An error occurred! try:     print(a)  except:     print(d) finally:     print("You can't print an undefined variable") </pre>
try, except	Try a block of code, else, raise a defined exception	
finally	Execute a final code when the try and the except blocks fail	<pre> output: You can't print an undefined variable a=7+2 if a&lt;10:     raise Exception("Oh! You didn't get a score of 10") </pre>
raise Exception()	Raise an exception that stops the command when execution isn't possible	
import x	Import a whole module or library	import math
from x import y	Import a library x from a file, or a class y	from scipy.stats import mode
as	Customize an expression to your preferred name	<pre> import pandas as pd x=[1, 4, 6, 7] if 5 in x:     print("There is a five") else:     print("There is no five") </pre>
in	Check if a value is present in a variable	<pre> output: There is no five x=[1, 4, 6, 7] x=b print(x is b) True </pre>
is	Check if two variables refer to a single element	
None	Declare a null value	<pre> x=None 5&lt;10 </pre>
<	Check if one value is lesser than another	<pre> output: True 5&gt;10 </pre>
>	Check if one value is more than another	<pre> output: False 2*2&lt;=3 </pre>
<=	Check if a value is lesser or equal to another	<pre> output: False 2*2&gt;=3 </pre>
>=	Check if a value is greater or equal to another	<pre> output: True </pre>

"=="	Check if a value is exactly equal to the other	3==4 output: False
!="	Ascertain that a value is not equal to the other	3!=4 output: True
import re	Import Python's built-in regular expressions	import re re.findall("strings", variable)
a b	Check if either of two elements are present in a string	import re someText="Hello regular expression" a=re.findall("regular Hello", someText) print(a) output: ['Hello', 'regular']
string\$	Check if a variable ends with a set of strings	import re someText="Hello regular expression" a=re.findall("expression\$", someText) print(a) output: ['expression']
^string	Check if a variable starts with a set of strings	import re someText="Hello regular expression" a=re.findall("^Hello", someText) print(a) output: ['Hello']
string.index()	Check the index position of a string character	a= "Hello World" a.index('H') output: 0
string.capitalize()	Capitalize the first character in a set of strings	a= "Hello World" a.capitalize() output: 'Hello world'
string.swapcase()	Print the first letter of each word as a lower case and the others as upper case	a= "Hello World" a.swapcase() output: 'hELLO wORLD'
string.lower()	Convert all the strings to a lowercase	a= "Hello World" a.lower() output: 'hello world'
string.upper()	Convert all strings to uppercase	a= "Hello World" a.upper() output: 'HELLO WORLD'
string.startswith()	Check if a string starts with a particular character	a= "Hello World" a.startswith('a') output: False
string.endswith()	Check if a string ends with a particular character	a= "Hello World" a.endswith('d') output: True
string.split()	Separate each word into a list	a= "Hello World" a.split() output: ['Hello', 'world']
strings {}.format()	Display an output as string	a=3+4 print("The answer is {}".format(a)) output: The answer is 7
is not None	Check if the value of a variable is not empty	def checknull(a): if a is not None: return "its full!" else: return "its empty!" 9%4
x%y	Find the remainder (modulus) of a division	output: 1

		9//4
x//y	Find the quotient of a division	output: 2
"="	Assign a value to a variable	a={1:5, 3:4} ["a two"] + ["a one"]  output: ['a two', 'a one']
		1+3
"+"	Add elements together	output=4 3-4
"_"	Find the difference between a set of numbers	output=-1 3*4
"*"	Find the product of a set of numbers	output:12 a=2 a+=3
a+=x	Add x to variable a without assigning its value to a new variable	output: 5 a=3 a-=2
a-=x	Subsract x from variable a without assigning it to a new variable	output: 1 a=[1, 3, 4] a*=2
a*=x	Find the product of variable a and x without assigning the result to a new variable	output: [1, 3, 4, 1, 3, 4] 2**3
x**y	Raise base x to power y	output: 8 pow(2, 3)
pow(x, y)	Raise x to the power of y	output: 8 abs(-5)
abs(x)	Convert a negative integer to its absolute value	output: 5 8**(1/3)
x**(1/nth)	Find the nth root of a number	output: 2
a=b=c=d=x	Assign the same value to multiple variables	a=b=c=d="Hello world" x = [1, 2] y = 3 x, y = y, x print(x, y)
x, y = y, x	Swap variables	output: 3 [1, 2] a=[1, 3, 5] for b in a: print(b, "x", "2", "=", b*2)
for	Loop through the elements in a variable	output: 1 x 2 = 2 3 x 2 = 6 5 x 2 = 10 a=4 b=2 while b<=a: print(b, "is lesser than", a) b+=1
while	Keep looping through a variable, as far as a particular condition remains True	output: 2 is lesser than 4 3 is lesser than 4 4 is lesser than 4

		<pre>x=range(4) print(x) range(0, 4) for b in x: print(b)</pre>
		<pre>output: 0 1 2 3</pre>
range()	Create a range of positive integers between x and y	<pre>print(sum([1, 2, 3]))</pre>
sum()	Iterate through the elements in a list	<pre>output:6 print(sum([1, 2, 3], 3))</pre>
sum(list, start)	Return the sum of a list with an added element	<pre>output: 9</pre>
[]	Make a list of elements	<pre>x=['a', 3, 5, 'h', [1, 3, 3], {'d':3}]</pre>
()	Create a tuple---tuples are immutable	<pre>x=(1, 2, 'g', 5)</pre>
{}	Create a dictionary	<pre>a={'x':6, 'y':8} x=[1, 3, 5, 6] x[0:2]</pre>
x[a:b]	Slice through a list	<pre>output: [1, 3] a={'x':6, 'y':8} print(a['x'])</pre>
x[key]	Get the value of a key in dictionary x	<pre>output: 6 x=[1] x.append([1,2,3]) print(x)</pre>
x.append()	Add a list of values to an empty list	<pre>output: [1, [1,2,3]] x=[1,2] x.extend([3,4,6,2]) print(x)</pre>
x.extend()	Add a list of values to continue an existing list without necessarily creating a nested list	<pre>output: [1, 2, 3, 4, 6, 2] x=[1,2,3,5] del(x[0:2]) print(x)</pre>
del(x[a:b])	Delete an item completely from a list at a specific index	<pre>output: [2,3,5] y={1:3, 2:5, 4:6, 8:2} del(y[1], y[8]) print(y)</pre>
del(x[key])	Delete a key and a value completely from a dictionary at a specific index	<pre>output= {2:5, 4:6} a={1:3, 2:4, 5:6} a.pop(1)</pre>
dict.pop()	Pop out the value of a key and remove it from a dictionary at a specific index	<pre>output: 3 a={1:2, 4:8, 3:5} a.popitem()</pre>
dict.popitem()	Pop out the last item from a dictionary and delete it	<pre>output: (3, 5) print(a) output: {1:2, 4:8} a=[1, 3, 2, 4, 1, 6, 6, 4] a.pop(-2)</pre>
list.pop()	Pop out a given index from a list and remove it from a list	<pre>output: 6 print(a) output: [1, 3, 2, 4, 1, 6, 4] x=[1, 3, 5] x.clear() print(x)</pre>
clear()	Empty the elements of a list or a dictionary	<pre>output: [] x=[1, 5, 6, 7] x.remove(1)</pre>
remove()	Remove an item from a list	<pre>output: [5, 6, 7]</pre>

		<pre>x=[3, 5, 6] x.insert(1, 4) print(x)</pre>
<code>insert()</code>	Insert elements into a llist	output: [1, 4, 3, 5, 6]
		<pre>x=[1, 3, 5, 6] x.sort(reverse=True) print(x)</pre>
<code>sort(reverse=condition)</code>	Reverse the direction of the elements in a list	output: [6, 5, 3, 1]
		<pre>x={1:3, 5:6} x.update({1:4, 8:7, 4:4}) print(x)</pre>
<code>update()</code>	Update a dictionary by changing its first element and adding any other item to its end	output: {1: 4, 5: 6, 8: 7, 4: 4}
		<pre>a={1:2, 4:8} a.keys()</pre>
<code>keys()</code>	Show all the keys in a dictionary	output: dict_keys([1, 4])
		<pre>a={1:2, 4:8} a.values()</pre>
<code>values()</code>	Show all the values in a dictionary	output: dict_values([2, 8])
		<pre>a={1:2, 4:8} a.items()</pre>
<code>items()</code>	Display the keys and the values in a dictionary	output: dict_items([(1, 2), (4, 8)])
		<pre>a={1:2, 4:8, 3:5} a.get(1)</pre>
<code>get(key)</code>	Get the value of an item in a dictionary by its key	output: 2
<code>setdefault(key)</code>	Return the original value of an element to a dictionary	<pre>a.setdefault(2) a={'x':6, 'y':8} b={'c':5, 'd':3} f={'**a, **y} print(f)</pre>
<code>f={'**a, **b}</code>	Merge two dictionaries	output: {'x': 6, 'y': 8, 'c': 5, 'd': 3}
		<pre>a=[1, 3, 2, 4, 4, 1, 6, 6, 4] a.remove(4) print(a)</pre>
<code>remove()</code>	Remove the first matching value of an element from a list without minding its index	output: [1, 3, 2, 4, 1, 6, 6, 4]
<code>memoryview(x)</code>	Access the internal buffers of an object	<pre>a=memoryview(object)</pre>
<code>bytes()</code>	Convert a memory buffer protocol into bytes	<pre>bytes(a[0:2])</pre>
<code>bytearray()</code>	Return an array of bytes	<pre>bytearray(object)</pre>
<code>#</code>	Write a single line of comment or prevent a line of code from being executed	<pre># Python regex cheat sheet """The Python regex cheat sheet is good for beginners It's equally a great refresher for experts"""</pre>
<code>""" """</code>	Write a multi-line comment	

# Command Line

pip install package	Install an online library	pip install pandas
virtualenv name	Use virtualenv to create a virtual environment	virtualenv myproject
mkvirtualenv name	Use virtual environment wrapper to create virtual environment	mkvirtualenv myproject
python file.py	Run the commands in a Python file	"python my_file.py
pip freeze	List out all the installed packages in a virtual environment	pip freeze
pip freeze > somefiles	Copy all installed libraries in a single file	pip freeze > requirements.txt
where	Find the installation path of Python	where python
--version	Check the version of a package	python --version
.exe	Run a Python shell	python.exe
with open(file, 'w')	Write to an existing file and overwrite its existing content	with open('regex.txt', 'w') as wf: wf.write("Hello World!")
with open(file, 'r')	Open a file as read-only	with open('regex.txt', 'r') as rf: print(rf.read())
with open(file, 'a')	Write to a file without overwriting its existing content	with open('regex.txt', 'a') as af: af.write("\nHello Yes!") af=open('regex.txt')
file.close	Close a file if it's not in use	af.close
exit	Exit the Python shell	exit()