Python RegEx Cheat Sheet

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

elif:	Make a counterstatement when the if statement is False	if 2<3:
		print("Two is smaller")
		elif 2==3:
		print("Go on")

else:	Make a final counterstatement if other conditions are False	if 2<3:
		print("Two is smaller")
		elif 2==3:
		print("Go on")
		else:
		print("Three is greater")

continue	Ignore a condition and execute the rest of the loop	a=[1, 4, -10, 6, 8]
		for b in a:
		if b<=0:
		continue
		print(b)
		
		output:
		1
		4
		6
		8

break	Terminate the flow of a loop with a given condition	a=[1, 4, -10, 6, 8]
		for b in a:
		if b>=6:
		break
		print(b)
		
		output:
		1
		4
		-10

pass	Ignore a set of prior instructions	for b in a: pass
try, except	Try a block of code, else, raise a defined exception	try: print(a) except: print("An error occured!") output: An error occured!
finally	Execute a final code when the try and the except blocks fail	try: print(a) except: print(d) finally: print("You can't print an undefined variable") output: You can't print an undefined variable
raise Exception()	Raise an exception that stops the command when execution isn't possible	a=7+2 if a<10: raise Exception("Oh! You didn't get a score of 10")
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	import pandas as pd

in	Check if a value is present in a variable	x=[1, 4, 6, 7]
		if 5 in x:
		print("There is a five")
		else:
		print("There is no five")
		
		output: There is no five

is	Check if two variables refer to a single element	x=[1, 4, 6, 7] x=b print(x is b) True
None	Declare a null value	x=None
<	Check if one value is lesser than another	5<10 output: True
>	Check if one value is more than another	5>10 output: False
<=	Check if a value is lesser or equal to another	2*2<=3 output: False
>=	Check if a value is greater or equal to another	2*2>=3 output: True

"==	Check if a value is exactly equal to the other	3==4 ouput: False
!=	Ascertain that a value is not equal to the other	3!=4 ouput: 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) 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!"

x%y	Find the remainder (modulus) of a division	9%4 output: 1

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

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

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

Command Line

pip install package	Install an online library	pip install pandas
virtualenv name	Use virtaulenv 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!")
file.close	Close a file if it's not in use	af=open('regex.txt') af.close
exit	Exit the Python shell	exit()