

Python – Zusammenfassung

Eingebaute Funktionen

<code>__import__()</code>	<code>classmethod()</code>	<code>format()</code>	<code>issubclass()</code>	<code>open()</code>	<code>sorted()</code>
<code>abs()</code>	<code>compile()</code>	<code>frozenset()</code>	<code>iter()</code>	<code>ord()</code>	<code>staticmethod()</code>
<code>all()</code>	<code>complex()</code>	<code>getattr()</code>	<code>len()</code>	<code>pow()</code>	<code>str()</code>
<code>any()</code>	<code>delattr()</code>	<code>globals()</code>	<code>list()</code>	<code>print()</code>	<code>sum()</code>
<code>ascii()</code>	<code>dict()</code>	<code>hasattr()</code>	<code>locals()</code>	<code>property()</code>	<code>super()</code>
<code>bin()</code>	<code>dir()</code>	<code>hash()</code>	<code>map()</code>	<code>range()</code>	<code>tuple()</code>
<code>bool()</code>	<code>divmod()</code>	<code>help()</code>	<code>max()</code>	<code>repr()</code>	<code>type()</code>
<code>breakpoint()</code>	<code>enumerate()</code>	<code>hex()</code>	<code>memoryview()</code>	<code>reversed()</code>	<code>vars()</code>
<code>bytearray()</code>	<code>eval()</code>	<code>id()</code>	<code>min()</code>	<code>round()</code>	<code>zip()</code>
<code>bytes()</code>	<code>exec()</code>	<code>input()</code>	<code>next()</code>	<code>set()</code>	
<code>callable()</code>	<code>filter()</code>	<code>int()</code>	<code>object()</code>	<code>setattr()</code>	
<code>chr()</code>	<code>float()</code>	<code>isinstance()</code>	<code>oct()</code>	<code>slice()</code>	

Reguläre Ausdrücke

Flags <code>re.ASCII</code> nur ASCII-Zeichen bei <code>\w</code> , <code>\b</code> , <code>\s</code> und <code>\d</code> <code>re.DOTALL</code> . passt auf alle Zeichen, auch <code>\n</code> <code>re.IGNORECASE</code> Gross-/Kleinschreibung ignorieren <code>re.MULTILINE</code> betrifft die Metazeichen <code>^</code> und <code>\$</code> <code>re.VERBOSE</code> übersichtliche Formatierung möglich		Wiederholungen <code>*</code> 0 oder mehrmals <code>+</code> 1 oder mehrmals <code>?</code> 0 oder einmal <code>{m,n}</code> m- bis n-mal	
Zeichenklassen <code>\d</code> Dezimalziffer <code>[0-9]</code> <code>\D</code> keine Dezimalziffer <code>[^0-9]</code> <code>\s</code> Leer-/Steuerzeichen <code>[\t\n\r\f\v]</code> <code>\S</code> kein Leer-/Steuerzeichen <code>[^\t\n\r\f\v]</code> <code>\w</code> Unicode Wortzeichen <code>[a-zA-Z0-9_]</code> (inkl. Uml.) <code>\W</code> kein Wortzeichen <code>[^a-zA-Z0-9_]</code>		Prüfzeichen <code> </code> <code>x y</code> passt entweder auf <code>x</code> oder auf <code>y</code> <code>^</code> Anfang des Strings (oder Zeile mit <code>re.MULTILINE</code>) <code>\$</code> Ende des Strings (oder Zeile mit <code>re.MULTILINE</code>) <code>\A</code> Anfang des Strings <code>\Z</code> Ende des Strings <code>\b</code> Wortgrenze <code>\B</code> keine Wortgrenze	
Übereinstimmungen <code>re.match(pattern, string, flags=0)</code> <code>re.search(pattern, string, flags=0)</code> <code>re.findall(pattern, string, flags=0)</code> <code>re.finditer(pattern, string, flags=0)</code>		Modifizierungen <code>re.split(pattern, string, maxsplit=0, flags=0)</code> <code>re.sub(pattern, repl, string, count=0, flags=0)</code> <code>re.subn(pattern, repl, string, count=0, flags=0)</code>	
Gruppierung <code>(Ausdruck)</code> Normale Gruppierung <code>(?P<Name>Ausdruck)</code> Benannte Gruppierung <code>(?:Ausdruck)</code> Passive Gruppierung		Annahmen <code>(?=Ausdruck)</code> positiv, vorausschauend <code>(?!Ausdruck)</code> negativ, vorausschauend <code>(?<=Ausdruck)</code> positiv, zurückschauend <code>(?<!Ausdruck)</code> negativ, zurückschauend	

String-Methoden

Überprüfungen <code>str.isalnum()</code> <code>str.isalpha()</code> <code>str.isascii()</code> <code>str.isdecimal()</code> <code>str.isdigit()</code> <code>str.isidentifier()</code> <code>str.islower()</code> <code>str.isnumeric()</code> <code>str.isprintable()</code> <code>str.isspace()</code> <code>str.istitle()</code> <code>str.isupper()</code> <code>str.startswith(prefix[, start[, end]])</code> <code>str.endswith(suffix[, start[, end]])</code>		Formatierungen <code>str.center(width[, fillchar])</code> <code>str.ljust(width[, fillchar])</code> <code>str.rjust(width[, fillchar])</code> <code>str.zfill(width)</code> <code>str.format(*args, **kwargs)</code> <code>str.format_map(mapping)</code>	
		Umwandlungen <code>str.encode(encoding="utf-8", errors="strict")</code> <code>str.expandtabs(tabsize=8)</code> <code>str.replace(old, new[, count])</code> <code>str.strip([chars])</code> <code>str.rstrip([chars])</code> <code>str.lstrip([chars])</code>	
Gross-/Kleinschreibung <code>str.capitalize()</code> <code>str.casefold()</code> <code>str.lower()</code> <code>str.upper()</code> <code>str.swapcase()</code> <code>str.title()</code>	Teilen & Kombinieren <code>str.join(iterable)</code> <code>str.partition(sep)</code> <code>str.rpartition(sep)</code> <code>str.split(sep=None, maxsplit=-1)</code> <code>str.rsplit(sep=None, maxsplit=-1)</code> <code>str.splitlines([keepends])</code>	Suchen <code>str.count(sub[, start[, end]])</code> <code>str.find(sub[, start[, end]])</code> <code>str.rfind(sub[, start[, end]])</code> <code>str.index(sub[, start[, end]])</code> <code>str.rindex(sub[, start[, end]])</code>	

String-Format

format_spec	[[fill]align][sign][#][0][width][grouping_option][.precision][type]		
fill	<any character>		
align	"<" ">" "=" "^"		
sign	"+" "-" " "		
width	digit+		
grouping_option	"_" ","		
precision	digit+		
type	"b" "c" "d" "e" "E" "f" "F" "g" "G" "n" "o" "s" "x" "X" "%"		

List

list.append(x)	list.count(x)	list.insert(i, x)	list.reverse()
list.clear()	list.extend(iterable)	list.pop([i])	list.sort(key=None, reverse=False)
list.copy()	list.index(x[, start[, end]])	list.remove(x)	

NumPy (import numpy as np)

Array erzeugen		
np.empty(shape[, dtype, order])	np.array(object[, dtype, copy, order, subok, ndmin])	
np.empty_like(prototype[, dtype, order, subok])	np.asarray(a[, dtype, order])	
np.eye(N[, M, k, dtype, order])	np.copy(a[, order])	
np.identity(n[, dtype])	np.fromfile(file[, dtype, count, sep, offset])	
np.ones(shape[, dtype, order])	np.fromstring(string[, dtype, count, sep])	
np.ones_like(a[, dtype, order, subok, shape])	np.loadtxt(fname[, dtype, comments, delimiter])	
np.zeros(shape[, dtype, order])	np.arange([start,] stop[, step,][, dtype])	
np.zeros_like(a[, dtype, order, subok, shape])	np.linspace(start, stop[, num, endpoint])	
np.full(shape, fill_value[, dtype, order])	np.logspace(start, stop[, num, endpoint, base])	
np.full_like(a, fill_value[, dtype, order])	np.meshgrid(*xi, **kwargs)	
Array-Manipulation		
np.reshape(a, newshape[, order])	np.concatenate((a1, a2, ...)[, axis, out])	
ndarray.flatten([order])	np.stack(arrays[, axis, out])	
np.ravel(a[, order])	np.column_stack(tup)	
np.unravel_index(indices, shape[, order])	np.split(ary, indices_or_sections[, axis])	
np.transpose(a[, axes]), auch ndarray.T	np.array_split(ary, indices_or_sections[, axis])	
np.atleast_1d(*arys)	np.tile(A, reps)	
np.atleast_2d(*arys)	np.repeat(a, repeats[, axis])	
np.atleast_3d(*arys)	np.append(arr, values[, axis])	
np.expand_dims(a, axis)	np.roll(a, shift[, axis])	
Lineare Algebra		
np.dot(a, b)	np.linalg.inv(a)	np.linalg.norm(x, ord=None, axis=None, keepdims=False)
np.matmul(x1, x2)	np.linalg.solve(a, b)	
M @ v	np.cross(a, b)	

Matplotlib (import matplotlib.pyplot as plt)

Erstellen			
plt.figure([num, figsize, dpi, facecolor, ...])		plt.twinx([ax])	
plt.subplot(*args, **kwargs)		plt.tight_layout([pad, h_pad, w_pad, rect])	
plt.subplots([nrows, ncols, sharex, sharey, ...])		plt.savefig(*args, **kwargs)	
plt.twinx([ax])		plt.show(*args, **kw)	
Zeichnen		Dekorieren	
plt.annotate(s, xy, *args, **kwargs)		plt.axis(*args, **kwargs)	
plt.arrow(x, y, dx, dy, **kwargs)		plt.colorbar([mappable, cax, ax])	
plt.contour(*args[, data])		plt.clabel(CS, *args, **kwargs)	
plt.contourf(*args[, data])		plt.grid([b, which, axis])	
plt.loglog(*args, **kwargs)		plt.legend(*args, **kwargs)	
plt.plot(*args[, scalex, scaley, data])		plt.margins(*margins[, x, y, tight])	
plt.scatter(x, y[, s, c, marker, cmap, norm, ...])		plt.suptitle(t, **kwargs)	
plt.semilogx(*args, **kwargs)		plt.title(label[, fontdict, loc, pad])	
plt.semilogy(*args, **kwargs)		plt.xlabel(xlabel[, fontdict, labelpad])	
plt.stem(*args[, linefmt, markerfmt, basefmt, ...])		plt.xlim(*args, **kwargs)	
plt.step(x, y, *args[, where, data])		plt.xticks([ticks, labels])	
plt.streamplot(x, y, u, v[, density, linewidth, ...])		plt.ylabel(ylabel[, fontdict, labelpad])	
plt.text(x, y, s[, fontdict, withdash])		plt.ylim(*args, **kwargs)	
		plt.yticks([ticks, labels])	
Linestyle	Marker	Color	
"-", "solid"	"."=point	"o"=circle	"v"=triangle_down
":", "dotted"	"s"=square	"p"=pentagon	">"=triangle_right
--", "dashed"	"*"=star	"d", "D"=diamond	"^"=triangle_up
"-.", "dashdot"	"x", "X"=x	"+", "P"=plus	"<"=triangle_left
		"b", "blue"	"g", "green"
		"c", "cyan"	"m", "magenta"
		"w", "white"	"y", "yellow"
			"r", "red"
			"k", "black"
			"#0F0F0F"
			(0.0, 0.5, 1.0)

SciPy

scipy.interpolate.interp1d(x, y, kind='linear', axis=-1, copy=True, bounds_error=None, assume_sorted=False)
scipy.optimize.bisect(f, a, b, args=(), maxiter=100, full_output=False, disp=True)
scipy.misc.derivative(func, x0, dx=1.0, n=1, args=(), order=3)
scipy.integrate.trapz(y, x=None, dx=1.0, axis=-1)