

Creative Computing Python Reference

Useful general-purpose functions

<pre>help(<search term>)</pre>	<p>Displays help text including examples about a function, type or keyword</p>	<pre>help(list)</pre> <p>see methods and usage examples for type list</p> <pre>help('for')</pre> <p>see usage of the for statement</p> <pre>help(len)</pre> <p>see usage for the len function</p>
<pre>print <expression></pre> <pre>print(<expression>)</pre>	<p>Displays the result of the expression on the console. Can be used as a function or a keyword (with or without parentheses). More flexible as a keyword. When used as a function, parameter must be a string.</p>	<pre>print 10 + 15</pre> <p>displays 25</p> <pre>print(10 + 15)</pre> <p>error: 25 is not a string</p> <pre>print(str(10 + 15))</pre> <p>displays 25</p> <pre>print "Result: ", 10</pre> <p>displays the text "Result: 10"</p>
<pre>len(<object>)</pre>	<p>Returns the number of items in a sequence (list or string, for example)</p>	<pre>name = "Hanifa"</pre> <pre>print(len(name))</pre>

Turtle Graphics (requires from turtle import *)

<pre>forward(<pixels>)</pre> <pre>fd(<pixels>)</pre>	<p>Move the turtle forward by the specified number of pixels in the current orientation.</p>	<pre>forward(100)</pre> <p>Moves the turtle forward by 100 pixels</p>
<pre>backward(<pixels>)</pre> <pre>bk(<pixels>)</pre>	<p>Move the turtle backward by the specified number of pixels in the current orientation</p>	<pre>backward(100)</pre> <p>Moves the turtle backward by 100 pixels</p>
<pre>left(<angle>)</pre> <pre>lt(<angle>)</pre>	<p>Turn the turtle left by the specified number of degrees</p>	<pre>left(90)</pre> <p>Turns the turtle left 90 degrees</p>
<pre>right(<angle>)</pre> <pre>rt(<angle>)</pre>	<p>Turn the turtle right by the specified number of degrees</p>	<pre>right(90)</pre> <p>Turns the turtle right 90 degrees</p>
<pre>up()</pre>	<p>Bring the pen up (moving the turtle doesn't draw on the canvas)</p>	<pre>up()</pre> <pre>goto(0, 0)</pre> <p>Return to origin without leaving a trace.</p>
<pre>down()</pre>	<p>Bring the pen down (moving the turtle draws on the canvas)</p>	
<pre>hideturtle()</pre>	<p>Hide the turtle.</p>	
<pre>goto(<x>, <y>)</pre>	<p>Move the turtle to the specified location.</p>	<pre>goto(0, 0)</pre> <pre>down()</pre>

		goto(100, 100) Draw a line between the origin and the point (100, 100)
color(<str>) color(<red>, <green>, <blue>)	Change the drawing color according to either a string or red, green blue values between 0 and 1.	color("red") Makes the pen color red. color(1, 0, 1) Make the pen color purple.
bgcolor(<str>) bgcolor(<red>, <green>, <blue>)	Change the background color according to either a string or red, green blue values between 0 and 1.	Same as color
fill(<boolean>)	Change the fill status. Call fill(True) before drawing a closed shape to fill and fill(False) at the end.	color("green") fill(True) for i in range(4): forward(100) right(90) fill(False) Draw a green filled square.
shape(<shape>)	Change the turtle's appearance. By default, the following shapes are available: "arrow", "turtle", "circle", "square", "triangle", "classic"	shape("turtle") Changes the shape of the turtle to look like a turtle.
register_shape(<file>)	Add a GIF image as a possible turtle. Must be in the same directory.	register_shape("cake.gif") shape("cake.gif") Adds a cake image to the list of possible turtles and sets the turtle to look like a cake.
stamp()	Draw an impression of the turtle.	

Random (requires from random import *)

random()	Returns a float in the interval [0,1) (includes 0 but not 1)	print random() Could print, for example, 0.596742123314 if(random() < .5): print "get pizza" else: print "get teriyaki" Simulates randomly choosing between pizza and teriyaki.
randint(<min>, <max>)	Returns an integer in the range [min, max] (including both end points)	print randint(10, 15) Could print, for example, 12 rand = randint(0,5) count = 0 while(rand != 4): print "Not a 4, roll again!" rand = randint(0,5) count = count + 1

		<pre>print "got a 4 in", count, "tries" Simulates rolling a die until a 4 comes up. Displays how many rolls were made</pre>
<pre>choice(<sequence>)</pre>	<p>Returns a random element from a sequence (list or string)</p>	<pre>word = "computer" rand_let = choice(word) Chooses a random letter from the word "computer" and stores it in the rand_let variable. meal_opts = ["burgers", "sushi", "lasagna", "burritos", "onion soup"] print "I will eat", choice(meal_opts), "tonight." Randomly chooses a meal to have.</pre>

Type list

<pre>.append(<object>)</pre>	<p>Add <object> to the end of the list. <object> can be of any type.</p>	<pre>meal_opts = ["burgers", "sushi", "lasagna", "burritos", "onion soup"] meal_opts.append("mud") Adds the string "mud" to the pre- existing list.</pre>
<pre>.count(<value>)</pre>	<p>Returns the number of occurrence of a particular value.</p>	<pre>flips = ["H", "T", "T", "T", "H"] print flips.count("H") Prints the number of times "H" occurs in the list (2 in this case).</pre>
<pre>.index(<value>)</pre>	<p>Returns the first index of a value.</p>	<pre>flips = ["H", "T", "T", "T", "H"] print flips.index("T") Prints the index of the first occurrence of "T" (1 in this case).</pre>
<pre>.pop()</pre>	<p>Removes and returns the last item.</p>	<pre>meal_opts = ["burgers", "sushi", "lasagna", "burritos", "onion soup"] last = meal_opts.pop() print last Prints and removes "onion soup"</pre>
<pre>.pop(<index>)</pre>	<p>Removes and returns the item at the specified index.</p>	
<pre>.remove(<value>)</pre>	<p>Removes the first occurrence of <value>.</p>	<pre>players = ["Jenn", "Cedric", "Julie"] players.remove("Jenn") print players Prints ['Cedric', 'Julie']</pre>
<pre>.reverse()</pre>	<p>Reverses the list. This modifies the existing list.</p>	<pre>players.reverse() players now holds ['Julie', 'Cedric']</pre>

<code>.sort()</code>	Sorts the list alphanumerically. This modifies the existing list	<pre>vals = [1, 10, 5, "ten", "fifteen", "apple"] vals.sort() vals now holds [1, 5, 10, 'apple', 'fifteen', 'ten']</pre>
----------------------	--	--

Type str (strings)

<code>.upper()</code>	Returns a copy of the string all in uppercase. Does NOT modify the string.	<pre>text = "be quiet" print text.upper() Displays "BE QUIET"</pre> <pre>text.upper() print text Displays "be quiet" because the call on upper did not modify the string.</pre>
<code>.lower()</code>	Returns a copy of the string all in lowercase. Does NOT modify the string.	<pre>print "heY yOu".lower() Displays "hey you"</pre>
<code>.capitalize()</code>	Returns a copy of the string with only the first character capitalized. Does NOT modify the string.	<pre>print "heY yOu".capitalize() Displays "Hey you"</pre>
<code>.title()</code>	Returns a copy of the string with the first letter of each token (space separated string of characters) capitalized. Does NOT modify the string.	<pre>print "heY yOu".title() Displays "Hey You"</pre>
<code>.find(<str>)</code>	Return the lowest index at which the substring is found. Return -1 if not found.	<pre>print "hey you".find("y") Displays 2.</pre>
<code>.index(<str>)</code>	Like find, but throws error if not found.	<pre>print "hey you".index("y") Displays 2.</pre>
<code>.count(<str>)</code>	Returns the number of times a substring appears in the string.	<pre>name = "Lakayla" print name.count("a") Displays 3.</pre> <pre>cheer = "Go Garfield! Go Garfield! Go Garfield!" print cheer.count("Garfield") Displays 3</pre>
<code>.endswith(<str>)</code>	Return True if the string ends with the suffix, False otherwise.	<pre>word = "ducks" if(word.endswith("s")): print "plural" else: print "singular" Uses the last letter of a word to determine whether or not it's plural (not very well, though!)</pre>
<code>.startswith(<str>)</code>	Returns True if the string begins with the prefix, False otherwise.	<pre>name = raw_input("Your name? ") if(name.startswith("Dr. ")):</pre>

		<pre>print "Marry me? "</pre> <p>Asks the user for their name and asks them for marriage if their name starts with Dr.</p>
<code>.isalpha()</code>	Returns True if all characters in the string are alphabetic and there is at least one character in the string, False otherwise.	<pre>word = raw_input("Give me a word: ") while(not word.isalpha()): word = raw_input("That wasn't a word! Try again: ")</pre> <p>Asks the user for input until the full string is composed of letters.</p>
<code>.isdigit()</code>	Returns True if all characters in the string are digits and there is at least one character in the string, False otherwise.	<pre>guess = raw_input("Guess a number: ") if(not guess.isdigit()): print "Come on, that's not a number!"</pre> <p>Asks the user for a number and tells them if they didn't input a number.</p>
<code>.join(<seq>)</code>	Concatenates all elements of the sequence (list), separated by the string and returns it.	<pre>title = ["Fear", "and", "Loathing", "in", "Las", "Vegas"] print "-".join(title)</pre> <p>Displays "Fear-and-Loathing-in-Las-Vegas"</p>
<code>.split(<str>)</code>	Return a list of the words in the string using str as the delimiter or whitespace if str is not specified.	<pre>title = "Fear-and-Loathing-in-Las-Vegas" words = title.split("-") words now contains the list ["Fear", "and", "Loathing", "in", "Las", "Vegas"]</pre>
<code>.strip()</code>	Returns a copy of the string with leading and trailing whitespace removed.	<pre>text = " It was a dark and stormy night " print text.strip()</pre> <p>Displays "It was a dark and stormy night"</p>
<code>.replace(<str>, <str2>)</code>	Replaces every instance of <str> replaced by <str2> and returns the resulting string.	<pre>last_name = "Schmitz" last_name = last_name.replace("itz", "ortz")</pre> <p>Displays "Schmortz"</p>

Files

<code>open(<filename>)</code>	Given a filename as a string, returns an open file.	<pre>records = open("data.txt") for line in records: print line</pre> <p>prints every line in the data.txt file.</p>
-------------------------------------	---	--