

Python reference sheet

Classes and types

NoneType	None
bool	False
int	5040
float	3.141
str	"Escape \"double-quotes\", not 'single'"
str	'Escape \'single-quotes\' , not "double"'
list	[360,'abc',3.141,360]

Input

Store input in variable *value* `value = type(input(prompt))`

Libraries and built-in functions

Here is a selection of the functions available. You may use others.

Turtle library

move forward <i>distance</i> steps	<code>turtle.fd(steps)</code>
move backward <i>distance</i> steps	<code>turtle.bk(steps)</code>
turn <i>degrees</i> to the left(anti-clockwise)	<code>turtle.lt(degrees)</code>
turn <i>degrees</i> to the right (clockwise)	<code>turtle.rt(degrees)</code>
go to position <i>x</i> , <i>y</i>	<code>turtle.goto(x, y)</code>
set turtle <i>x</i> position	<code>turtle.setx(x)</code>
set turtle <i>y</i> position	<code>turtle.sety(y)</code>
pick pen up	<code>turtle.pu()</code>
put pen down	<code>turtle.pd()</code>
set pen colour	<code>turtle.color(colour)</code>
set fill colour	<code>turtle.fillcolor(colour)</code>
start filling the shape	<code>turtle.begin_fill()</code>
stop filling the shape	<code>turtle.end_fill()</code>

Built-in functions

round value to digits places	<code>round(value, digits)</code>
absolute value	<code>abs(value)</code>
smaller of two values	<code>min(value1, value2)</code>
larger of two values	<code>max(value1, value2)</code>
returns a new lowercase string	<code>str.lower()</code>
returns a new uppercase string	<code>str.upper()</code>
get length of string <i>str</i>	<code>len(str)</code>
returns True if string is all whitespace	<code>str.isspace()</code>
returns True if string is all alpha characters	<code>str.isalpha()</code>
returns True if string is all digits	<code>str.isdigit()</code>

Math library

rounds up	<code>math.ceil(value)</code>
rounds down	<code>math.floor(value)</code>
square root	<code>math.sqrt(value)</code>

Random library

return random number between `random.randint(start, end)`
start and *end* (inclusive)

Functions

define	<code>def fun(parameters):</code> <i>code-block</i> return
define with optional parameter, <i>p2</i>	<code>def fun(p1, p2=5):</code> <i>code-block</i> return
calling	<code>fun(parameters)</code>
assigning result of a function to <i>var</i>	<code>var = fun(parameters)</code>

Flow control

if statement	<code>if condition:</code> <i>code-block</i>
if-else statement	<code>if condition1:</code> <i>code-block</i> <code>else:</code> <i>code-block</i>
if-elif-else statement	<code>if condition1:</code> <i>code-block</i> <code>elif condition2:</code> <i>code-block</i> <code>else:</code> <i>code-block</i>
while statement	<code>while condition:</code> <i>code-block</i>
for statement (count)	<code>for i in range(n):</code> <i>code-block</i> # from 0 to n
for statement	<code>for v in values:</code> <i>code-block</i>

List comprehensions

[*expr* for *element* in *list*]
[*expr* for *element* in *list* if *condition* ...]

Lists

append <i>x</i> to the end of list <i>xs</i>	<code>xs.append(x)</code>
append list <i>b</i> the end of list <i>xs</i>	<code>xs.extend(b)</code>
insert <i>x</i> in position <i>i</i> of the list <i>xs</i>	<code>xs.insert(i, x)</code>
remove the first occurrence of <i>x</i> in <i>xs</i>	<code>xs.remove(x)</code>
remove then return the last element of <i>xs</i>	<code>xs.pop()</code>
remove then return the <i>i</i> -th element of <i>xs</i>	<code>xs.pop(i)</code>
index of the first occurrence of <i>x</i>	<code>xs.index(x)</code>
between <i>i</i> and <i>j</i>	<code>xs.index(x,i,j)</code>
count occurrences of <i>x</i> in <i>xs</i>	<code>xs.count(x)</code>
sort the list <i>xs</i>	<code>xs.sort()</code>
reverse the list <i>xs</i>	<code>xs.reverse()</code>

Strings

returns length of a string <i>s</i>	<code>len(s)</code>
returns True if substring <i>sbstr</i>	
is in the string <i>s</i>	<code>sbstr in s</code>
converts a string <i>s</i> into upper case	<code>s.upper()</code>
converts a string <i>s</i> into lower case	<code>s.lower()</code>
returns True if all characters in the	
string <i>s</i> are numeric	<code>s.isdigit()</code>
returns a string where a specified	
value "xy" is replaced with a specified	
value "ab"	<code>s.replace("xy", "ab")</code>

String formatting

format <i>x</i> to string using format	<code>"{0}".format(x)</code>
format <i>x</i> to string using f string	<code>f'Let's print {x}'</code>