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

Built-in functions

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

Math library

rounds up	`math.ceil(value)`
rounds down	`math.floor(value)`
square root	`math.sqrt(value)`

Random library

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

Functions

define

```
def fun(parameters):
    code-block
    return
```

define with optional parameter, *p2*

```
def fun(p1, p2=5):
    code-block
    return
```

calling

```
fun(parameters)
```

assigning result of a function to *var*

```
var = fun(parameters)
```

Flow control

if statement

```
if condition:
    code-block
```

if-else statement

```
if condition1:
    code-block
else:
    code-block
```

if-elif-else statement

```
if condition1:
    code-block
elif condition2:
    code-block
else:
    code-block
```

while statement

```
while condition:
    code-block
```

for statement (count)

```
for i in range(n):
    code-block    # from 0 to n
```

for statement

```
for v in values:
    code-block
```

List comprehensions

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

Lists

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

Strings

returns length of a string s	`len(s)`
returns True if substring sbstr is in the string s	`sbstr in s`
converts a string s into upper case	`s.upper()`
converts a string s into lower case	`s.lower()`
returns True if all characters in the string s are numeric	`s.isdigit()`
returns a string where a specified value "xy" is replaced with a specified value "ab"	`s.replace("xy", "ab")`
returns the number of times a specified value occurs in a string	`s.count("xy")`

String formatting

format *x* to string using `format`	`"{0}".format(x)`
format *x* to string using f string	`f'Let's print {x}'`