Some useful Python functions:

- \texttt{abs(x)} returns the absolute value of \( x \).
- \texttt{m \% n} returns the remainder when integer \( m \) is divided by positive integer \( n \).
- \texttt{m // n} corresponds to integer division when \( n \) and \( m \) are both integers (that is, the quotient).
- \texttt{min(...) \text{ returns the minimum value among its arguments (which may be numbers, strings, or a list).}}
- \texttt{max(...) \text{ returns the maximum value among its arguments (which may be numbers, strings, or a list).}}
- \texttt{print(x, end='\n')} prints \( x \) on a current line, followed by \( 'n' \) (or followed by \( 'n' \) if \( end \) is not specified).
- \texttt{check.expect(comment, value1, value2)} prints a summary message indicating if the test passed (if \( \text{value1} == \text{value2} \)) or failed.
- \texttt{check.within(comment, value1, value2, tol)} prints a summary message indicating if the test passed (if \( \text{abs(value1 - value2)} \leq \text{tol} \)) or failed.
- \texttt{check.set_input(x1, x2, ..., xn)} sets up program to use parameters when \text{input} is called by a tested function.
- \texttt{check.set_screen(s)} prints \( s \) as expected output when a test is run.
- \texttt{check.set_print_exact(x1, x2, ..., xn)} checks parameters against printed output when test is run.
- \texttt{type(x)} returns the type of \( x \). For example, you may check if \( x \) is of type \text{Bool} with \( \text{type(x)==type(True)} \).
- \text{Type conversion operations include:} \texttt{str(x), int(x), float(x), list(x)}.

String functions and methods in Python:

- \texttt{len(s)} returns the number of characters in \( s \).
- \texttt{s[a:b]} returns a string containing the characters at positions \( a, a+1, ..., b-1 \) for \( 0 \leq a \leq b \leq \text{len(s)} \). There is no error if \( b > \text{len(s)} \).
- \texttt{s[a:b:c]} returns a string containing the characters at positions \( a, a+c, a+2c, ... \). The last character in the new string comes before position \( b \) in \( s \).
- \texttt{s in t \text{ returns True if string } s \text{ occurs as a substring in } t, \text{ and False otherwise.}}
- \texttt{s + t \text{ returns a new string containing the characters of string } s \text{ followed by the characters of string } t}.\text{ \text{input}(p) \text{ returns a string entered by keyboard input after the prompt } p \text{ is displayed. Returned string does not include newline character.}}
- \texttt{s.count(c) \text{ returns the number of times string } c \text{ occurs in string } s \text{ (could be 0).}}
- \texttt{s.format(v0, v1, ...)} returns a string like \( s \), except that \( v0 \) replaces \{0\}, \( v1 \) replaces \{1\}, etc.
- \texttt{s.find(t) \text{ returns the index of the first occurrence of } t \text{ in } s \text{ (returns } -1 \text{ if } t \text{ is not a substring of } s).}}
- \texttt{s.isalnum()} returns \text{True} if \( s \) is nonempty and all characters are alphabetical (letters) or numeric (digits), and \text{False} if the string is empty or it is nonempty and at least one character is not alphabetical or numeric.
- \texttt{s.isdigit()} returns \text{True} if all characters in \( s \) are digits ('0', '9'), and \text{False} otherwise. \text{Returns False for the empty string.}}
- \texttt{s.islower()} returns \text{True} if all characters in \( s \) are lowercase, and \text{False} otherwise. \text{Returns False for the empty string.}}
- \texttt{s.isupper()} returns \text{True} if all characters in \( s \) are uppercase, and \text{False} otherwise. \text{Returns False for the empty string.}}
- \texttt{s.join(L), where \( L \) is a \text{(listof Str)} \text{, returns the string } L[0]+s+L[1]+s+...+s+L[-1].}}
- \texttt{s.lower()} returns a string like \( s \), except all uppercase characters are replace by lowercase versions.
- \texttt{s.replace(a, b) \text{ returns a new string like } s \text{, except that all occurrences of } a \text{ are replaced with } b.}}
- \texttt{s.split()} returns a list of strings from \( s \), by dividing \( s \) at whitespace. If \( s \) has value "my dog has fleas.\n", then \( s.split() \text{ returns ["my","dog","has","fleas."].}}
- \texttt{s.startswith(t) \text{ returns True if string } s \text{ begins with the string } t, \text{ and False otherwise.}}
- \texttt{s.strip()} returns a string like \( s \), but leading and trailing whitespace (including newline characters) are removed.
- \texttt{s.upper()} returns a string like \( s \), except all lowercase characters are replaced by uppercase versions.
List functions and methods in Python:

• `len(L)` returns the number of values in L.
• `sum(L)` returns the sum of all entries in L (must be numbers).
• `L[a:b]` returns the list `[L[a], L[a+1], ..., L[b-1]]` for `0 ≤ a ≤ b ≤ len(L)`. There is no error if `b > len(L)`.
• `L[a:b:c]` returns the list `[L[a], L[a+c], L[a+2*c], ...]`. The last item in the new list comes before position `b` in L.
• `list(map(func,lst))` returns the list that results from applying `func` to each element of `lst` (also works if `lst` is a string).
• `list(filter(func,lst))` returns the list of all elements of `lst` for which `func` returns `True` (also works if `lst` is a string).
• `x in L` returns `True` if `x` is an element of L, and `False` otherwise.
• `L+M` returns a new list containing the elements of the list L followed by the elements of the list M.
• `L.extend(M)` returns `None` and mutates the list L by adding the elements of list M to the end of list L.
• `L.append(x)` returns `None` and mutates the list L by placing the value x at the end of the list L.
• `L.index(x)` returns the smallest index `j` such that `L[j] = x` if `x` is in L, and results in an error if `x` is not in L.
• `L.insert(p,x)` returns `None` and mutates the list L by inserting `x` into position `p`, and keeping other values in L in the same relative positions.
• `L.remove(x)` returns `None` and mutates the list L by removing the first occurrence of the value `x`, and results in an error if `x` is not in L.
• `L.pop(k)` returns `L[k]` and mutates the list L by removing the value at position `k`, and results in an error if `k` is not a valid list position.
• `L.sort()` returns `None` and mutates the list L by sorting it into increasing order.
• `L.reverse()` returns `None` and mutates the list L by reversing the order of the elements.