DO NOT DETACH THIS PAGE
Short Python function/method descriptions:

```
__builtins__:
  input([prompt: str]) -> str
    Read a string from standard input. The trailing newline is stripped. The prompt string,
    if given, is printed without a trailing newline before reading.
  abs(x: float) -> float
    Return the absolute value of x.
  chr(i: str) -> Unicode character
    Return a Unicode string of one character with ordinal i; 0 <= i <= 0x10ffff.
  float(x: object) -> float
    Convert x to a floating point number, if possible.
  int(x: object) -> int
    Convert x to an integer, if possible.  A floating point argument will be truncated
    towards zero.
  len(x: object) -> int
    Return the length of the list, tuple, dict, or string x.
  max(iterable: object) -> object
  max(a, b, c, ...) -> object
    With a single iterable argument, return its largest item.
    With two or more arguments, return the largest argument.
  min(iterable: object) -> object
  min(a, b, c, ...) -> object
      With a single iterable argument, return its smallest item.
      With two or more arguments, return the smallest argument.
  open(name: str[, mode: str]) -> TextIO
    Open a file.  Legal modes are "r" (read) (default), "w" (write), and "a" (append).
  ord(c: str) -> int
    Return the integer ordinal of a one-character string.
  print(value: object, ..., sep=' ', end='\n') -> None
    Prints the values. Optional keyword arguments:
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.
  range([start: int], stop: int, [step: int]) -> list-like-object of int
    Return the integers starting with start and ending with stop - 1 (positive step)
    or stop + 1 (negative step), with step specifying the amount to increment (or decrement).
    If start is not specified, the list starts at 0.  If step is not specified,
    the values are incremented by 1.

dict:
  D[k] --> object
    Produce the value associated with the key k in D.
  del D[k]
    Remove D[k] from D.
  k in D --> bool
    Produce True if k is a key in D and False otherwise.
  D.get(k: object) -> object
    Return D[k] if k in D, otherwise return None.
  D.keys() -> list-like-object of object
    Return the keys of D.
  D.values() -> list-like-object of object
    Return the values associated with the keys of D.
  D.items() -> list-like-object of Tuple[object, object]
    Return the (key, value) pairs of D, as 2-tuples.
```

DO NOT DETACH THIS PAGE

```
file open for reading (TextIO):
  F.close() -> None
    Close the file.
  F.read() -> str
    Read until EOF (End Of File) is reached, and return as a string.
  F.readline() -> str
    Read and return the next line from the file, as a string. Retain any newline.
    Return an empty string at EOF (End Of File).
  F.readlines() -> List[str]
    Return a list of the lines from the file.  Each string retains any newline.

file open for writing (TextIO):
  F.close() -> None
    Close the file.
  F.write(x: str) -> int
    Write the string x to file F and return the number of characters written.

list:
  x in L --> bool
    Produce True if x is in L and False otherwise.
  L.append(x: object) -> None
    Append x to the end of the list L.
  L.extend(iterable: object) -> None
    Extend list L by appending elements from the iterable. Strings and lists are
    iterables whose elements are characters and list items respectively.
  L.index(value: object) -> int
    Return the lowest index of value in L, but raises an exception if value does
    not occur in S.
  L.insert(index: int, x: object) -> None
    Insert x at position index.
  L.pop([index: int]) -> object
    Remove and return item at index (default last).
  L.remove(value: object) -> None
    Remove the first occurrence of value from L.
  L.reverse() -> None
    Reverse the list *IN PLACE*.
  L.sort() -> None
    Sort the list in ascending order *IN PLACE*.

str:
  x in s --> bool
    Produce True if x is in s and False otherwise.
  str(x: object) -> str
    Convert an object into its string representation, if possible.
  S.count(sub: str[, start: int[, end: int]]) -> int
    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are interpreted
    as in slice notation.
  S.endswith(S2: str) -> bool
    Return True if and only if S ends with S2.
  S.find(sub: str[, i: int]) -> int
    Return the lowest index in S (starting at S[i], if i is given) where the
    string sub is found or -1 if sub does not occur in S.
  S.index(sub: str) -> int
    Like find but raises an exception if sub does not occur in S.
```

DO NOT DETACH THIS PAGE

```
S.isalpha() -> bool
  Return True if and only if all characters in S are alphabetic
  and there is at least one character in S.
S.isdigit() -> bool
  Return True if all characters in S are digits
  and there is at least one character in S, and False otherwise.
S.islower() -> bool
  Return True if and only if all cased characters in S are lowercase
  and there is at least one cased character in S.
S.isupper() -> bool
  Return True if and only if all cased characters in S are uppercase
  and there is at least one cased character in S.
S.lower() -> str
  Return a copy of the string S converted to lowercase.
S.lstrip([chars: str]) -> str
  Return a copy of the string S with leading whitespace removed.
  If chars is given and not None, remove characters in chars instead.
S.replace(old: str, new: str) -> str
  Return a copy of string S with all occurrences of the string old replaced
  with the string new.
S.rstrip([chars: str]) -> str
  Return a copy of the string S with trailing whitespace removed.
  If chars is given and not None, remove characters in chars instead.
S.split([sep: str]) -> List[str]
  Return a list of the words in S, using string sep as the separator and
  any whitespace string if sep is not specified.
S.startswith(S2: str) -> bool
  Return True if and only if S starts with S2.
S.strip([chars: str]) -> str
  Return a copy of S with leading and trailing whitespace removed.
  If chars is given and not None, remove characters in chars instead.
S.upper() -> str
  Return a copy of the string S converted to uppercase.
```