Reading input from the user
\texttt{input}(message): Displays message to the user and returns what the user typed as a string

Reading from a file
\begin{verbatim}
with \texttt{open(filename, "r") as file:}
    for line in file:
        # do something with line (a string)
\end{verbatim}

\begin{itemize}
\item \textbf{Range}
\texttt{range}\,(\texttt{stop}): Equivalent to \texttt{range(0, stop, 1)}
\texttt{range}\,(\texttt{start, stop[, step]}): Create sequence from inclusive \texttt{start} to exclusive \texttt{end} by \texttt{step}
\item \textbf{Slicing}
\texttt{seq\,[\texttt{start}[:\texttt{end}[:\texttt{step}]]]}: Slice \texttt{seq} from inclusive \texttt{start} to exclusive \texttt{end} by \texttt{step}
\end{itemize}

\section*{Strings}

\begin{itemize}
\item The following functions are built-in and answer questions about strings
\texttt{len} (\texttt{string}): Returns the number of characters in the string
\texttt{int} (\texttt{string}), \texttt{float} (\texttt{string}): Converts a string to an int or float
\item String object methods
\texttt{upper}(), \texttt{lower}(), \texttt{capitalize}(): Returns a new upper or lower-cased, or 1st letter upper-cased string
\texttt{find} (\texttt{some\_string}): Returns the first index that \texttt{some\_string} occurs at in the string or -1 if not found
\texttt{find} (\texttt{some\_string, index}): Same as above, but starts searching at index
\texttt{replace} (\texttt{old, new}): Return a copy of the string with all occurrences of \texttt{old} substituted with new
\texttt{startswith} (\texttt{prefix}): Returns \texttt{True} if the string starts with \texttt{prefix}, \texttt{False} otherwise
\texttt{endswith} (\texttt{suffix}): Returns \texttt{True} if the string ends with \texttt{suffix}, \texttt{False} otherwise
\texttt{strip}(): Returns a copy of the string with leading and trailing whitespace removed
\texttt{split}(): Return a list of the words in the string using whitespace as the delimiter
\item String operators
\texttt{\texttt{string1 + string2}}: Returns a new string that is the concatenation of \texttt{string1} and \texttt{string2}
\texttt{\texttt{string * int}}: Returns a new string that is \texttt{string} repeated \texttt{int} times
\texttt{\texttt{substr in string}}: Returns \texttt{True} if \texttt{substr} is a substring of \texttt{string}, \texttt{False} otherwise
\end{itemize}

\section*{Lists}

\begin{itemize}
\item Creating new lists
\texttt{[]} creates empty list
\texttt{[\texttt{object1, object2, ...}]} creates list containing objects
\texttt{list} (\texttt{iterable}) creates a list from any iterable object (e.g., range, set, string)
\item The following functions are built-in and answer questions about lists
\texttt{len} (\texttt{list}): Returns the number of elements in \texttt{list}
\texttt{sum} (\texttt{list}), \texttt{min} (\texttt{list}), \texttt{max} (\texttt{list}): Returns the sum, min, or max of elements in \texttt{list}
\texttt{sorted} (\texttt{list}): Returns a new copy of the list in sorted order
\item List object methods
\texttt{append} (\texttt{x}): Adds \texttt{x} to the end of the list
\texttt{extend} (\texttt{other\_list}): Adds all elements of \texttt{other\_list} the end of the list
\texttt{index} (\texttt{item}): Returns the index of the first occurrence of \texttt{item} in the list or error otherwise
\texttt{insert} (\texttt{index, x}): Insert \texttt{x} at \texttt{index} in the list
\texttt{pop} (): Removes the item at the end of the list and returns it
\texttt{pop} (\texttt{index}): Removes item at \texttt{index} from the list and returns it
\texttt{reverse} (): Reverses the elements in the list
\texttt{sort} (): sorts the elements in the list
\item List operators
\end{itemize}
list1 + list2: Returns a new list that contains the elements of list1 followed by the elements of list2
list * int: Returns a new list that contains the items in list repeated int times
item in list: Returns True if item is an element of list, False otherwise

Sets

- Creating new sets
  set() creates empty set
  {elt1, elt2, ...} creates a new set with the given elements
  set(iterable) creates a set from any iterable object (e.g., string, list)
- The following functions are built-in and answer questions about sets
  len(set): Returns the number of elements in the set
- Set object methods
  add(elt): Adds elt to the set
  clear(): Removes all elements from the set
  pop(): Removes an arbitrary element from the set and returns it
  remove(elt): Removes elt from the set
- Set operators
  elt in set: Returns True if elt is an element of set, False otherwise
  set1 <= set2: Returns True if set1 is a subset of set2 (every element of set1 is in set2), False otherwise
  set1 | set2: Returns union of the two sets (new set with elements from both set)
  set1 & set2: Returns intersection of the two sets (new set with only elements common to both sets)
  set1 - set2: Returns set difference (new set with elements set1 not in set2)

Dictionaries

- Creating new dictionaries
  {} creates empty dictionary
  {key1:value1, key2:value2, ...} creates a new dictionary with key-value pairs
- The following functions are built-in and answer questions about dictionaries
  len(dict): Returns the number of entries (key-value pairs) in the dictionary
- Dictionary object methods
  clear(): Removes all entries from the dictionary
  keys(): Returns an iterable object of the keys in the dictionary
  values(): Returns an iterable object of the values in the dictionary
  items(): Returns an iterable object of all (key, value) tuples in the dictionary
  get(key[, item]): Returns value associated with key if in dictionary, item otherwise. item defaults to None.
  pop(key[, item]): Removes key if in dictionary returning the associated value, returns item otherwise.
- Dictionary operators
  item in dict: Returns True if item is in the keys of dict, False otherwise

Tuples

- Creating new tuples
  () creates empty tuple
  (object1, object2, ...) creates tuple containing objects
- The following functions are built-in and answer questions about tuples
  len(tuple): Returns the number of elements in the tuple
- Tuple operators
  item in tuple: Returns True if item is contained in tuple, False otherwise
  tuple1 + tuple2: Returns a new tuple that is the concatenation of tuple1 and tuple2