Name_____ NCSU Login ID _____

Short Python function/method descriptions:

builtins functions:

input([prompt]) -> 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) -> number
  Return the absolute value of x.
int(x) -> int
  Convert x to an integer, if possible. A floating point argument will be truncated
  towards zero.
enumerate(iterable) -> iterator for index, value of iterable
  Return an enumerate object consisting of index, value pairs.
len(x) -> int
  Return the length of the list, tuple, dict, or string x.
max(iterable) -> 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
min(a, b, c, ...) -> object
  With a single iterable argument, return its smallest item.
  With two or more arguments, return the smallest argument.
print(value, ..., sep=' ', end='\n') -> NoneType
  Prints the values. Optional keyword arguments:
  sep: string inserted between values, default a space.
  end: string appended after the last value, default a newline.
open(name[, mode]) -> file open for reading, writing, or appending
  Open a file. Legal modes are "r" (read), "w" (write), and "a" (append).
range([start], stop, [step]) -> list of integers
  Return the integers starting with start and ending with stop - 1 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.
zip(sequence1, sequence2, …) -> list of tuples
  Returns a list of tuples, where each tuple contains the i-th element from each of the argument
  sequences. The returned list is truncated in length to the length of the shortest argument sequence.

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.has_key(k) --> bool
   Produce True if k is a key in D and False otherwise.
D.get(k) -> 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 of (object, object)
  Return the (key, value) pairs of D, as 2-tuples.

file open for reading:
F.close() -> NoneType
  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 newline.
  Return an empty string at EOF (End Of File).
F.readlines() -> list of str
  Return a list of the lines from the file. Each string ends in a newline.

file open for writing:
F.close() -> NoneType
  Close the file.
F.write(x) -> 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) -> NoneType
  Append x to the end of the list L. *IN PLACE*.
L.extend(iterable) -> NoneType
  Extend list L by appending elements from the iterable. Strings and lists are
  iterables whose elements are characters and list items respectively. *IN PLACE*.
L.index(value) -> int
  Return the lowest index of value in L.
L.insert(index, x) -> NoneType
  Insert x at position index. *IN PLACE*.
L.pop() -> object
  Remove and return the last item from L. *IN-PLACE*.
L.remove(value) -> NoneType
  Remove the first occurrence of value from L. *IN-PLACE*.
L.reverse() -> NoneType
  Reverse. *IN PLACE*.
L.sort() -> NoneType
  Sort the list in ascending order. *IN-PLACE*.

str:
x in s --> bool
  Produce True if and only if x is in s.
str(x) -> str
  Convert an object into its string representation, if possible.
S.count(sub[, start[, end]]) -> 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.find(sub[, i]) -> 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) -> int
  Like find but raises an exception if sub does not occur in S.
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
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, new) -> str
Return a copy of string S with all occurrences of the string old replaced
with the string new.
S.rstrip([chars]) -> 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]) -> list of 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.strip([chars]) -> 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.

arcpy cheatsheet

arcpy functions

arcpy

messaging
← AddMessage(message)
← GetMessages( )

licensing
← CheckOutExtension(ext. code)

describing data
← Describe(data)

Create objects
← Array
← Point
← Polyline
← Polygon
← ValueTable

parameters
← GetParameterAsText(index)
← SetParameterAsText(index, arg)

lists
← ListFields(dataset,{wild_card}, {field_type})
← ListRasters({wild_card}, {geom_type})

tools
← Buffer_analysis(params…)
← sa.SquareRoot(params…)

exceptions
← ExecuteError

describe data objects

Describe
† baseName
† dataType
† extension

DataSet
† datasetType
† extent

FeatureClass
† featureType
† shapeType

RasterDataset
† bandCount
† format

Workspace
† workspaceType

cursors (data access)

arcpy.da
← SearchCursor(data, fields, {where}, …)
← UdpdateCursor(data, fields, {where}…)
← InsertCursor(data, fields)

SearchCursor
←next( )
←reset( )

UpdateCursor
←next( )
←reset( )
←updateRow(row)
←deleteRow( )

InsertCursor
← insertRow(row)

geometry tokens
SHAPE@
SHAPE@XY
SHAPE@AREA

other objects

Array
Field
† name
† type
Geometry
Result
† outputcount
← getOutput(index)
Point
Polyline
Polygon
SpatialReference(item)
ValueTable
←AddRow

mapping

arcpy.mapping
Layer
‡ visible
‡ name
MapDocument
← AddLayer(df,layer…)
← ListLayers(map, …)
←ExportToPNG(map, pic,…)

environment settings

arcpy.env
‡ overwriteOutput
‡ workspace

cheatsheet key

† Read only property
‡ Read/write property
← Method
Class
Package or module