Python Part 2 – Command Arguments, Input, and Output

|  |  |
| --- | --- |
| **stdin**  **sys.stdin** is stdin in Python.  In Linux,we can redirect a file to stdin when launching python3.  Older versions of PyCharm don't support redirection of stdin so we have to use **open().**  When EOF is encountered, readline() returns an empty string. | **Example 1**: read loop until eof using stdin  #! /usr/bin/python  import sys  file = sys.stdin  while True:  inputLine = file.readline() # reads one text line from the file  # check for no input (i.e., EOF)  if inputLine == "":  break  inputLine = inputLine.rstrip('\n') # remove the newline  print(inputLine)  file.close()  $ python3 ex1.py <input.txt |
| **Opening a file directly**  **open(***filename, fileMode***)** returns a file object where *fileMode* is  r read mode (read an existing file)  w write mode (create/overwrite an existing file)  a append mode (create/append to an existing file)  **Reading a single text line**  *fileObject.***readline**() returns a string which will contain a new line character (which may be omitted if the last line in the file doesn't end in a new line).  **Closing a file**  **close**() closes an open file. | **Example 2**: read loop until eof using file open  #! /usr/bin/python  import sys  file = open("input.txt", "r")  while True:  inputLine = file.readline() # reads one text line from the file  # check for no input (i.e., EOF)  if inputLine == "":  break  inputLine = inputLine.rstrip('\n') # remove the newline  print(inputLine)  file.close()  $ python3 ex2.py |
| **Command Arguments**  To obtain command arguments, import sys.  The list of command arguments is in the list, **sys.argv:**   * **len(sys.argv) -** returns the number of arguments * **sys.argv[0] -** program name * **sys.argv[1]** - first actual argument | **Example 3**: reading a file that was specified as a command argument  #! /usr/bin/python  import sys  print ("#args=", len(sys.argv))  if len(sys.argv) < 2:  print("filename needed as command argument")  sys.exit(1)  file = open(sys.argv[1], "r")  while True:  inputLine = file.readline() # reads one text line from the file  # check for no input (i.e., EOF)  if inputLine == "":  break  inputLine = inputLine.rstrip('\n') # remove the newline  print(inputLine)  file.close()  $ python3 ex3.py input.txt  #args= 2  line one  line two |
| **Reading the contents from files specified as command arguments**  We can read and print the contents of multiple files specified as command arguments. Notice we are using a slice of sys.argv beginning with argument 1. | **Example 4**: print each file based on filenames in the sys.argv  #! /usr/bin/python  *# Use sys.argv list for the command arguments # open each file, print its name, print each line* import sys print (sys.argv) for fileName in sys.argv[1:]:  file = open(fileName, "r")  print (fileName + ":")  while True:  inputLine = file.readline()  if inputLine == "":  break  inputLine = inputLine.rstrip('\n') # remove the newline  print ("\t", inputLine)  file.close()  $ python3 ex4.py input.txt input2.txt  input.txt:  line one  line two  input2.txt:  1st line  2nd line  3rd line |
| **Reading multiple lines at once**  ***fileObject.*read()** attempts to read the entire contents of a file as one large string  ***fileObject.*readLines()** reads the entire contents of a file placing each text line in a list which is returned. | **Example 5**: read an entire file into a string  >>> # open a file for read, read the entire file into a string  f = open(**"input1.txt"**, **"r"**)  entireStr = f.read()  **Example 6**: read an entire file into a list  >>> # open a file for read, read the entire file into a list  f = open(**"input2.txt"**, **"r"**)  entireM = f.readLines() |
| **Checking file existence, type, size**  The **os.path** and **os** modules are helpful in checking file existence, type, and size.  **os.path.isfile**(*fileName*)  returns True if the specified file exists and is a file  **os.path.isdir**(*fileName*)  returns True if the specified file exists and is a directory  **os.path.getsize**(*fileName*)  returns the size (in bytes) for the specified file  **os.listdir**(*directory*)  returns a list of strings representing the files in the specified directory | **Example 7**: *Use os.path and os to examine files in the directory* **import** os.path **import** os print(**"os.path.isfile for 'one.txt':"**,  os.path.isfile(**"one.txt"**)) print(**"os.path.isdir for 'one.txt':"**,  os.path.isdir(**"one.txt"**)) print(**"os.path.isdir for '../catall':"**,  os.path.isdir(**"../catall"**)) print(**"os.path.getsize for '../catall':"**,  os.path.getsize (**"../catall"**)) print(**"os.path.getsize for 'one.txt:"**,  os.path.getsize (**"one.txt"**)) print(**"os.listdir for '../catall':"**,  os.listdir(**"../catall"**)) |
| **Exercise:** example 4 doesn't check whether a file exists before opening it. If the command argument isn't a file, the program will abort. Fix it to check. | #! /usr/bin/python  *# Use sys.argv list for the command arguments # open each file, print its name, print each line* import sys print (sys.argv) for fileName in sys.argv[1:]:  ??check to see if it is a file  if not os.path.isfile (filename): // = false or somthn  continue   file = open(fileName, "r")  print (fileName + ":")  while True:  inputLine = file.readline()  if inputLine == "":  break  inputLine = inputLine.rstrip('\n') # remove the newline  print ("\t", inputLine)  file.close() |
| **print() Function**  With Python 3, print changed syntax from Python 2. Previously in Python 2, print was a statement with this syntax: print *listOfItems*  With Python 3, it is a function with a variable number of parameters surrounded by parentheses.  By default, the print() function prints a line to stdout. It has some keyword parameters:  end=*endStr* specify what is printed at the end of this print line. By default, it is a \n. To suppress a new line, use end=""  sep=*sepStr* specify what is printed between parameters on this print line. By default, this is a single space.  file=*fileObj* specify which file to use. By default, it is **stdout**.  Also see the **string % operator** in Python Part 1. | >>> # print numbers across a line with two characters per number  for i in range (1, 20):  print ("%2i"%(i), end="")  **1 2 3 4 5 6 7 8 910111213141516171819**  >>> # print numbers and their square across a line.  # Print a space at the end of each print  for i in range (1, 10):  print (i, i\*i, end=" ")  **1 1 2 4 3 9 4 16 5 25 6 36 7 49 8 64 9 81**  >>> # print numbers and their square across a line.  # Print a space at the end of each print. Print ".." as separators.  for i in range (1, 10):  print (i, i\*i, sep="..", end=" ")  **1..1 2..4 3..9 4..16 5..25 6..36 7..49 8..64 9..81** |
|  |  |
|  |  |

**©2018 Larry W. Clark,** UTSA CS students may make copies for their personal use