**S1.3 What is the difference between a compiler and an interpreter?**

Compilers and Interpreters are two strategies for obtaining runnable code (traditionally called as *machine code*) from a program written in some programming language, say, high-level source language.

Compilers translates a program written in *source language* to *target language.*

* Compilation can be slow because it is difficult to translate from a high-level source language to low-level languages
* *Target language*  of compilers is *machine code,* which the computer processor knows how to execute

Interpreter reads the program and does whatever computation code describes.

* Given a program, interpreter can start running it without the time spent to compile it
* Code is more portable to different hardware architectures (any hardware architecture to which interpreter has been ported)
* It is slower than hardware execution of same computation because interpreter has to do many operations to figure out what it is supposed to be doing

|  |  |  |
| --- | --- | --- |
| **No** | **Compiler** | **Interpreter** |
| 1 | Complete program is given as input to the compiler (in human readable format) | Interpreter takes single step as input |
| 2 | Intermediate object code is generated | No intermediate object code is generated |
| 3 | Conditional control statements execute faster | Conditional control statements execute slower |
| 4 | Comparatively, more memory is required | Memory requirement is less |
| 5 | Errors are displayed after entire program is checked | Errors are displayed for every instruction interpreted |
| 6 | Example: C Compiler | Example: Python |

**S1.6 Where is 123 stored when the following statement (x = 123) is executed by the Python interpreter?**

All the variables and their respective values are stored in the main memory of the computer. The Main Memory is used to store information that the CPU needs in a hurry. The main memory is nearly as fast as the CPU. But the information stored in the main memory vanishes when the computer is turned off.

**S2.5 Write a program which prompts the user for a Celsius temperature and converts the temperature to Fahrenheit and prints out the converted temperature.**

##########################

# Celsius to Fahrenheit #

##########################

celsius = float(raw\_input("Enter temperature in celsius: "))

fahrenheit = ((celsius \* 9.0 / 5.0 ) + 32.0)

print "Temperature is %f celsius or %f fahrenheit" % (celsius, fahrenheit)

**S3.3 Write a program to prompt for a score between 0.0 and 1.0. If the score is out of range print an error message.  Otherwise, print a grade using the following table:**

**>= 0.9 A**

**>= 0.8 B**

**>= 0.7 C**

**>= 0.6 D**

**< 0.6 F**

##################

# Score to Grade #

##################

score = float(raw\_input("Enter score: "))

# Assigning grade to respective scores

while score >= 0.0 and score <= 1.0:

if score >= 0.9:

grade = 'A'

elif score >= 0.8:

grade = 'B'

elif score >= 0.7:

grade = 'C'

elif score >= 0.6:

grade = 'D'

else:

grade = 'F'

print "Grade is %c" % grade

break

else:

print "Oops! Not a valid score."

**S4.7 Rewrite the grade program above using a function called computegrade() that takes a score as its parameter and returns a grade as a string.**

##############################

# Score to Grade - Function #

##############################

def computegrade(score):

while score >= 0.0 and score <= 1.0:

if score >= 0.9:

grade = 'A'

elif score >= 0.8:

grade = 'B'

elif score >= 0.7:

grade = 'C'

elif score >= 0.6:

grade = 'D'

else:

grade = 'F'

return grade

break

else:

grade = "NA! Not a valid score entered."

return grade

score = float(raw\_input("Enter score: "))

print "Grade is " + computegrade(score)

**S5.1-2 Write a program which repeatedly reads numbers until the user enters ‘done’. Once ‘done’ is entered, print out the total, count, average, min, and max of the numbers.** **If the user enters anything but a number, use try-except to detect the mistake, print an error message, and skip to the next number.**

############################

# Calculating Basic Statistics #

############################

# Asking user for the entry and appending them to one list

num\_list = []

flag = 1

while flag == 1:

try :

num = raw\_input("Enter Number: ")

num = num.lower()

if num != 'done':

num\_list.append(float(num))

elif num == 'done':

flag = 0

except:

print "Wrong Entry! Enter a number or 'done'"

# Calculating and printing basic statistics for the final list

total = sum(num\_list)

count = len(num\_list)

average = total/count

min = min(num\_list)

max = max(num\_list)

print "Sum is %f" % total

print "Count is %f" % count

print "Average is %f" % average

print "Minimum is %f" % min

print "Maximum is %f" % max

**S6.5 Take the following Python code that stores a string**

str = ‘X-DSPAM-Confidence: 0.8475’

**and extract the portion of the string after the colon character. Then use float() to convert the extracted string into a floating point number.**

##################

# String Extract #

##################

str = "X-DSPAM-Confidence: 0.8475"

index\_of\_colon = str.find(':')

extracted\_string = str[(index\_of\_colon + 1) : len(str)]

string\_to\_num = float(extracted\_string.strip())

print "Extracted string (or floating point number) is ", string\_to\_num