Concepts of Programming Languages, CSCI 305, Fall 2021
Names, Scopes, and Bindings, Chapter 3, Nov. 3
Section 3.1 The Notion of Binding Time

1. Exercise 3.1, page 167
   Indicate the binding time (when the language is designed, when the program is linked, when the program begins execution, etc.) for each of the following decisions in your favorite programming language and implementation. Explain any answers you think are open to interpretation.

   a. The number of built-in functions (math, type queries, etc.)
      Language design time, although in some cases it might be expanded on during implementation (C started with a few built-in functions, such as sizeof. A large number of additional functions are defined by the standard library – printf, malloc, assert are often special-cased by the compiler in order to generate faster or safer code.)

   b. The variable declaration that corresponds to a particular variable reference (use)
      Say x is encountered in a procedure, Within different procedures, there may be several declarations of x. This question is asking which is used. The answer depends on whether the language uses static or dynamic scope.

      Static scope (C, C++, C#, Java, JavaScript, Pascal, started with Algol and most languages followed that) - the relevant declaration is determined "lexically" at compile time. Answer is compile time.

      Dynamic scope (early LISP, Common LISP and Scheme switched to static, but allow dynamic, Perl was dynamic, but now allows either) – the relevant declaration is determined at run time. Answer in this case would be run time.

   c. The maximum length allowed for a constant (literal) character string
      Likely to be at implementation time, however, it could be stated at language design time.

   d. The referencing environment for a subroutine that is passed as a parameter
      As question b, this depends on if the language uses static or dynamic scope:
         Static scope - referencing environment is determined at compile time.
         Dynamic scope - the referencing environment is determined at run time.

e. The address of a particular library routine

Link or load time. Could be run time in systems that perform dynamic linking. (Note that we are talking about virtual addresses; physical addresses are invisible to the running program, and are often changed by the operation system during execution.)

f. The total amount of space occupied by a program code and data

Run time since the amount of stack and heap space needed will often depend on the input.