SHA

Usage is very straightforward:

```
julia> using SHA

julia> bytes2hex(sha256("test"))
"9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08"
```

Each exported function (at the time of this writing, SHA-1, SHA-2 224, 256, 384 and 512, and SHA-3 224, 256, 384 and 512 functions are implemented) takes in either an `Array{UInt8}`, a `ByteString` or an `IO` object. This makes it trivial to checksum a file:

```
shell> cat /tmp/test.txt
test
julia> using SHA

julia> open("/tmp/test.txt") do f
           sha2_256(f)
       end
32-element Array{UInt8,1}:
 0x9f
 0x86
 0xd0
 0x81
 0x88
 0x4c
 0x7d
 0x65
    ⋮
 0x5d
 0x6c
 0x15
 0xb0
 0xf0
 0x0a
 0x08
```

Note the lack of a newline at the end of `/tmp/text.txt`. Julia automatically inserts a newline before the `julia>` prompt.

Due to the colloquial usage of `sha256` to refer to `sha2_256`, convenience functions are provided, mapping `shaxxx()` function calls to `sha2_xxx()`. For SHA-3, no such colloquialisms exist and the user must use the full `sha3_xxx()` names.

`shaxxx()` takes `AbstractString` and array-like objects (`NTuple` and `Array`) with elements of type `UInt8`.

Note that, at the time of this writing, the SHA3 code is not optimized, and as such is roughly an order of magnitude slower than SHA2.

« Random Numbers Serialization »