Variables and Data Types

Programming for Environmental Sciences

Variables

- A variable represents a memory location that is assigned a name.
- The memory location contains a value.
- We reference that value with the name assigned to that memory location.
- We can visualize variables, their names, and their values as shown

amount	36.84	volume	183.0
rate	0.065	total	486.5
temp	17.5	info	72

Variable Names

- Each variable must have a different name.
- Case sensitivity
  - case insensitive – uppercase and lowercase are equivalent
    - TEMP = temp
    - Fortran, IDL
  - case sensitive – uppercase and lowercase treated as distinct
    - TEMP not equal to temp
    - Python, R, unix, Matlab, passwords
- Special characters

Variable Names

- Rules
  - Fortran
    - Must start with a letter
    - After that, the rest of the name can contain only letters (a-z), digits (0-9) or underscore character _ (no blanks!)
    - Fortran 90 A variable name can be no longer than 31 characters.
    - Fortran77 6 characters
  - Python
    - Must start with a letter or an underscore
    - The remainder of your variable name may consist of uppercase and lowercase letters, numbers and underscores
    - Names are case sensitive
    - No length restriction

Variable Names

- Good Variable Names
  - Someone else may try to use your code
  - Well thought out names reduce the need for extensive comments
  - Make names descriptive temp better than t
  - Use multiple words temp_celcius lwp_modis
  - Separate words with underscore
  - The longer the name the more likely you are to make a typo
  - All lower case

Scientific Notation

- When a real number is very large or vary small, decimal notation does not work satisfactorily.

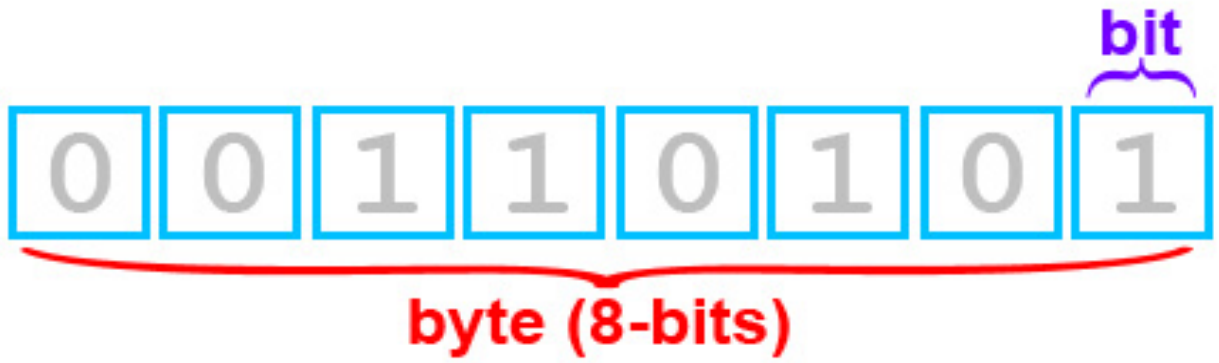
Decimal	Scientific	Exponential
3876000000	$3.876X10^{9}$	0.3876e10
0.0000010053	$1.0053X10^{-6}$	0.10053e-5
-8030000	$-8.03X10^{6}$	-.803e7

Data Types

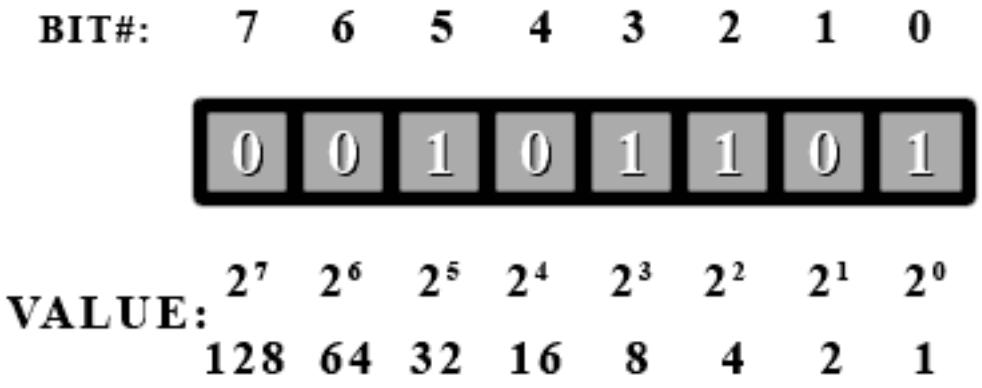
- Data typing help tells the processor what to expect and the amount of memory space to reserve for what to expect.

- A memory location can contain only one type of value
- Integer
  - No fractional portion and no decimal point
- Real
  - Contain a decimal point and may or may not have digits past the decimal point
  - Also called floating-point, float

A little binary

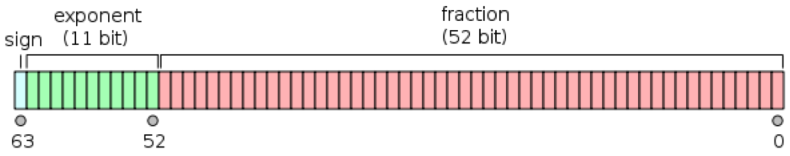



```
00000000 = 0
00000001 = 1
00000010 = 2
00000011 = 3
00000100 = 4
00000101 = 5
11111111 = 255
```



Data Types

Data Type	example	Storage size	Value range	Precision
Integer (short)	32	2 bytes	-32,768 to 32,767	
Integer (long)	2e9	4 bytes	-2,147,483,648 to 2,147,483,647	
Real	-15.45	4 bytes	1.2E-38 to 3.4E+38	6 decimal places
Double	3.1415926536	8 bytes	2.3E-308 to 1.7E+308	15 decimal places
Character	V	1 byte	0 to 255	
Logical	.TRUE.		.FALSE.	



The memory format of a double floating point value

How to define in different languages

- Explicit
  - Must declare every variable before you use it
  - Some languages let you choose, visual basic
  - REAL :: temperature, pressure Fortran90

- Implicit
  - You can use variables without declaring them
  - You don't have to declare anything ahead of time, because in these languages variables don't have types, values have types, and variables are just nametags attached to values, therefore there's nothing to declare about the type of a variable in the absence of a value.
  - Python, IDL, etc

Truncation

- When an arithmetic operation is performed using two real numbers, it's intermediate result is a real value
- Similarily, arithmetic operations between two integers yield an integer
- Length=side*3.5 side is real, length is integer
- The real result is stored in an integer
- When the computer stores a real number in an integer variable, it ignores the fractional portion and stores only the whole number portion of the real number.
- This type of rounding is called *truncation*

Computations with integers

- Mean=(n1+n2)/2
- All integers, result will be integer n1=2,n2=4,mean=3
- N1=2, n2=3, mean=2 instead of 2.5, all integers
- Ave=(n1+n2)/2 ave is real
- The result of integer arithmetic is still integer, the result is stored in a real value.
- Ave=(n1+n2)/2.0 all reals
- With rounding, the result is the integer closest in value to the real number
- With truncation, any decimal portion is dropped
- 15/8=1.875 rounded? Truncated?

Rounding Error

- Roundoff error is the difference between an approximation of a number used in computation and its exact (correct) value.
- Digital computers have size and precision limits on their ability to represent numbers
- Certain numerical manipulations are highly sensitive to roundoff errors.
- Large computations
- Adding a large and small number

Rounding Error

- Unfortunately, most decimal fractions cannot be represented exactly as binary fractions. A consequence is that, in general, the decimal floating-point numbers you enter are only approximated by the binary floating-point numbers actually stored in the machine.
- The problem is easier to understand at first in base 10. Consider the fraction 1/3. You can approximate that as a base 10 fraction:0.3, or better 0.33 or still better 0.333 and so on. No matter how many digits you're willing to write down, the result will never be exactly 1/3, but will be an increasingly better approximation of 1/3.
- In the same way, no matter how many base 2 digits you're willing to use, the decimal value 0.1 cannot be represented exactly as a base 2 fraction. In base 2, 1/10 is the infinitely repeating fraction
- 0.0001100110011001100110011001100110011001100110011...
- Stop at any finite number of bits, and you get an approximation.

Mixed Mode

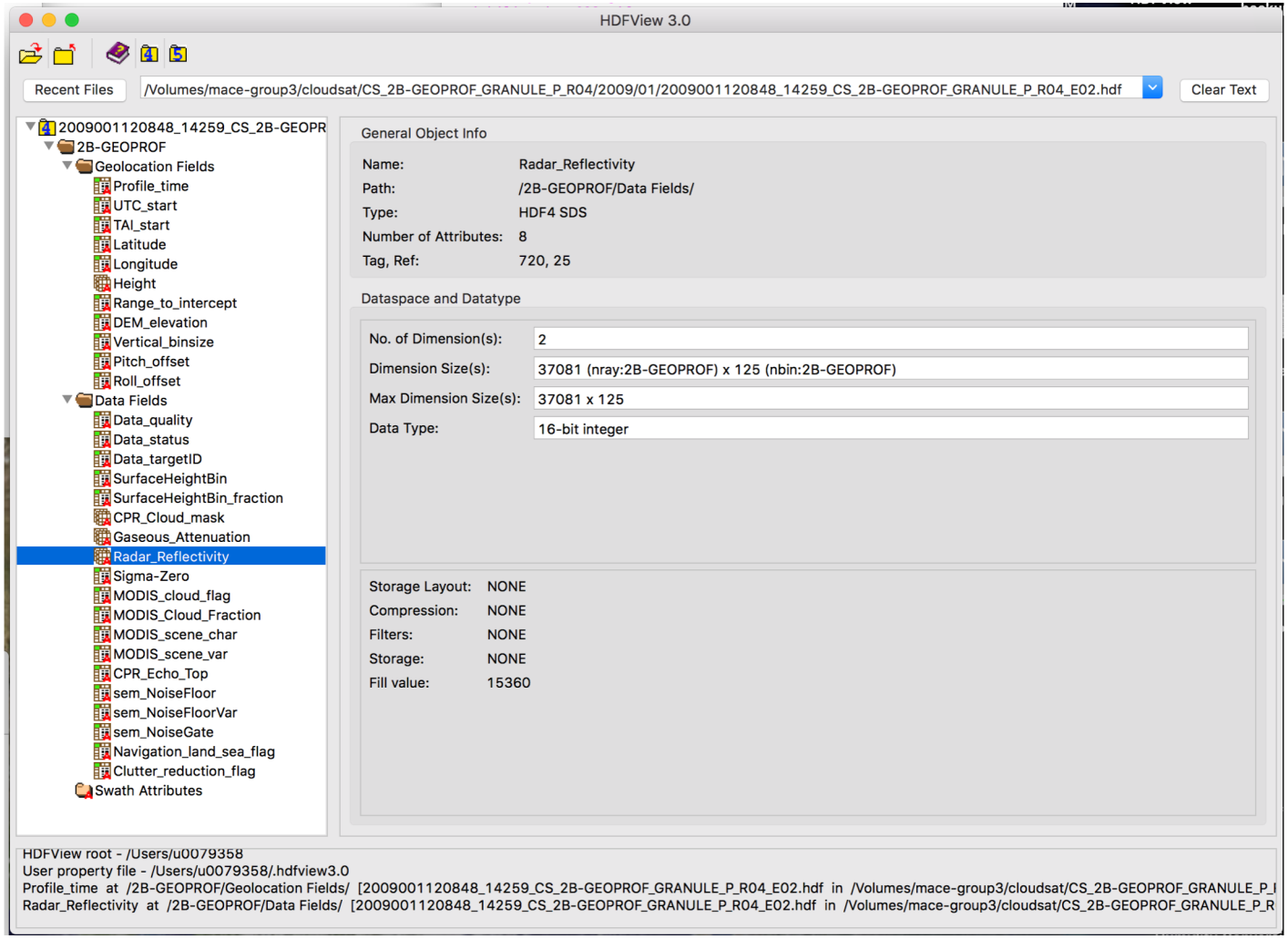
- Some programming languages take the simple way out and prohibit any expressions involving values of different data types and then provide special type conversion functions. This is called **strong typing** and such languages are called strongly typed.
- The C language is not strongly typed and instead infers the required type conversions from the context.

Time – number of seconds

- To search within a time array, you need something monotonically increasing and unique
- Seconds
  - Choose a base time
    - 1/1/1970 0:0:0 ,midnight, 1/1/1993 0:0:0, file start
  - A time offset is an amount of time subtracted from or added to the base time to get the time value of the data point
  - sbsmetM1.b1.20110425.000000.cdf
    - base_time = 1303689600
    - time_offset = 0, 60, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720, 780, 840, 900, 960, 1020, 1080, 1140, 1200, 1260, 1320, 1380, 1440, 1500,

Time – fractional day

- Fractional Day (julian day)
  - The Julian calendar, established by Julius Caesar in the year 45 BCE, the calendar specifies that every 4 years is a leap year, except if the year ends in a "00" then it is not a leap year, unless it is also divisible by 400 (in which case it is a leap year).
- Go to IDL program


HDFView 3.0
Recent Files
/Volumes/mace-group3/cloudsat/CS_2B-GEOPROF_GRANULE_P_R04/2009/01/2009001120848_14259_CS_2B-GEOPROF_GRANULE_P_R04_E02.hdf
Clear Text
2009001120848_14259_CS_2B-GEOPR
2B-GEOPROF
Geolocation Fields
Profile_time
UTC_start
TAI_start
Latitude
Longitude
Height
Range_to_intercept
DEM_elevation
Vertical_binsize
Pitch_offset
Roll_offset
Data Fields
Data_quality
Data_status
Data_targetID
SurfaceHeightBin
SurfaceHeightBin_fraction
CPR_Cloud_mask
Gaseous_Attenuation
Radar_Reflectivity
Sigma-Zero
MODIS_cloud_flag
MODIS_Cloud_Fraction
MODIS_scene_char
MODIS_scene_var
CPR_Echo_Top
sem_NoiseFloor
sem_NoiseFloorVar
sem_NoiseGate
Navigation_land_sea_flag
Clutter_reduction_flag
Swath Attributes
General Object Info
Name: Radar_Reflectivity
Path: /2B-GEOPROF/Data Fields/
Type: HDF4 SDS
Number of Attributes: 8
Tag, Ref: 720, 25
Dataspace and Datatype
No. of Dimension(s): 2
Dimension Size(s): 37081 (nray:2B-GEOPROF) x 125 (nbin:2B-GEOPROF)
Max Dimension Size(s): 37081 x 125
Data Type: 16-bit integer
Storage Layout: NONE
Compression: NONE
Filters: NONE
Storage: NONE
Fill value: 15360
HDFView root - /Users/u0079358
User property file - /Users/u0079358/.hdfview3.0
Profile_time at /2B-GEOPROF/Geolocation Fields/ [2009001120848_14259_CS_2B-GEOPROF_GRANULE_P_R04_E02.hdf in /Volumes/mace-group3/cloudsat/CS_2B-GEOPROF_GRANULE_P_R
Radar_Reflectivity at /2B-GEOPROF/Data Fields/ [2009001120848_14259_CS_2B-GEOPROF_GRANULE_P_R04_E02.hdf in /Volumes/mace-group3/cloudsat/CS_2B-GEOPROF_GRANULE_P_R

Overflow and Underflow

- Overflow
  - Result too large to store in the computer
- Underflow
  - The exponent of the result is too small to store in the computer
- If you get exponent underflow or overflow errors when you run your programs, you need to examine the magnitude of the values you are using.
- Go to IDL program