Variables, Expressions, Conditions, and All That

Variables

- Variables are placeholders for *locations in memory.*
  - Variables always have a *type* even if you don't have to declare it
  - The *primitive types* correspond (more or less) to the types defined in hardware, specifically integers, floating-point (single and double), and characters.
  - Many languages define more basic types including Booleans, strings, complex numbers, and so forth.

Types

- The computer has distinct **types** that are internally quite distinct. Each type has a set of **operators** defined on it.
  - Dynamic typing
    - Interpreter determines it by context
    - Primitive types are not declared at all
  - NumPy (introduced later) allows or requires some declaration of types

Numeric Types: Integers

- Integer
  - Quantities with no fractional part
  - Represented by sign bit + value in *binary*
    - *Computers do not use base 10 internally*
    - Both Fortran and Python use integers of size 32 bits
  - Maximum integer is is 2^{32}-1 (signed)
  - Python only
    - Long integer: has infinite precision
      - Computed in software and SLOW
      - *Not* the same as a "long" in C/C++

Floating Point

- Floating point single precision
  - Sign, exponent, mantissa
  - 32 bits in nearly all languages
  - IEEE 754 defines representation and operations
  - Approximately 6-7 decimal digits of precision, *approximate* exponent range is 10^{-126} to 10^{127}

Double

- Double precision floating point
  - Sign, exponent, mantissa
  - 64 bits
    - Number of bits NOT a function of the OS type! It is specified by the IEEE 754 standard!
  - Approximately 12-13 decimal digits of precision, approximate exponential range 10^{-1022} to 10^{1023}
- In Python you can assume that all floating-point numbers are doubles unless there was an explicit declaration somewhere

Complex

- Python supports complex types.
- A complex number consists of 2 doubles
  - `R+I*1J`
  - `R+I*1j`
  - It accepts either `J` or `j` (so not case sensitive in this context) but the numerical value of the imaginary part *must* immediately precede it. If the imaginary part is a variable as in these examples, the `1` must be present.

Non-Numeric Types: String

- String length can be dynamically determined but once set, it is fixed.
- Strings are *immutable* and cannot be changed. They can only be overwritten.

Boolean

- Boolean
  - Values can be True or False (note capitalization)
  - Are really integers but this is not important to the programmer

Variables and Literals

- Literals aka constants
  - Specified values e.g.

    3

    3.2

    10000000000000000L (Python long integer)

    "This is a string"

    True

    1.0+2J (Python complex)
- Variables
  - Have a type but the value must be assigned
  - Variables are assigned *locations in memory* by the compiler or interpreter

Type Conversions

- If a variable is of one type but it needs to be of a different type, it is necessary to do a *type conversion* aka a *cast*.
- An expression with more than one numeric type is said to be *mixed*.

```
N=20*3.5/11.0
```

- Most interpreters will automatically cast numeric variables to make mixed expressions consistent. The variables are promoted according to their rank. Lowest to highest the types are integer, float, double, complex.
- Almost no languages can or will automatically cast non-numeric types to numerics.

Type Conversions (Continued)

- Explicit casting among numeric types

```
R=float(I)
I=int(R)
Z=complex(r1,r2)
```

Numeric ⇔ Non-Numeric

- Python
  - It's straighforward

```
iage=39
age=str(iage)
age2='51'
iage=int(age2)
```

Most input methods read only strings (this is true of languages like Fortran/C as well but they do an internal conversion, Python usually does not)! Thus you must convert to numeric when appropriate, using one of the conversion functions.

Arithmetic Operators

- Operators defined on integers and doubles
- + - add subtract
- * / multiply divide
- ** exponentiation (can also use pow(a,b))
- Operator Precedence is:
- ** (* /) (+ -)
- All languages evaluate left to right by precedence unless told otherwise with parentheses

Integer Operators

- Python:
  - Integer division //
  - Remainder (mod) %

    2//3 = 0

    7%2=1
- Python Gotcha: versions below 2.7:

  2/3=0

  Versions 2.7 and above:

  2/3=.66666666666663

Logical/Boolean Operators

- Negation
  - not

    not flag
- AND
  - and
- OR
  - or

NonNumeric Operators

- Strings/Characters
  - There are many (some of which require function calls)
  - Concatenation +

    ```
    S1+S2
    ```

  - Substring extraction
  - S[0:3] WARNING: First character is counted as 0, and the last one in the substring is UB-1. This is characters 0, 1, and 2 of the string S.

Conditional Operators

- Conditional operators represent *relationships*. They can be defined on any type. Most commonly we use numerical conditional operators.
- These compare two numerical values for equality, non-equality, greater than, less than, greater than or equal to, less than or equal to.

Conditional Operator Precedence

- Like arithmetic operators, conditionals have a precedence. This may be somewhat language dependent but an example might be:
- greater/less outrank equal
- equal outranks and
- and outranks or

Comparison Operators

- Python
  - ==
  - !=
  - < > <= >=

Expressions

- Expressions are combinations of variables, literals, and operators and/or functions that can be evaluated to yield a value of one of the legal types.
- Examples

```
a+3*c
sqrt(abs(a-b))
A or B
```

Conditional Expressions

- Conditional expressions evaluate to *true* or *false*.

- `x > 2.0`
- `y > 0.0 and y < 1.0`
- `N == 0 or N == 1`

Statements

- A statement is one complete “sentence” in the language. It contains one complete instruction. Examples:
- assign A to B
  - In nearly all languages this is written

    ```
    B=A
    ```

    - Some older languages used := for assignment (to distinguish it from equality)

    ```
    B:=A
    ```

- Compute something and assign to a variable

  ```
  C=0.25*pi*d**2
  ```

Comments

- Python #
  - Anything to the right of # is ignored to the end of the line
  - Triple quotes “”” “””
  - Everything within triple quotes is treated as a literal string and a comment. It is intended for documentation strings (i.e. at the top of modules, etc.)

Making Choices

- Computer programs really can't do that many things. They can
  - Assign values to variables (memory)
  - Make decisions based on comparisons
  - Repeat a sequence of instructions over and over
  - Call subprograms
- Decisions are one of the fundamental programming constructs

Conditionals Cause Branching

- IF (comparison operation evaluating to Boolean) do something ELSE do something else
- IF (comparison) do something ELSE IF (comparison) do some other thing ELSE default behavior
- WARNING: In nearly all languages the comparison *short circuits*, i.e. once it determines T or F of the comparison it does not do any more evaluations. Don't rely on a compound comparison operation to evaluate a function or set a variable.

Python Syntax

elif and else are optional
No switch or case or equivalent, use elifs
Must use colons!

```
if comparison:
    code
elif comparison:
    more code
else:
    yet more code
```

Words to Symbols?

- From the Washington Post sports section, prior to the NFL playoffs. Conditional for Baltimore to win the division:

  Ravens clinch North with a win over the Bengals OR a tie and a Steelers tie or loss OR a Steelers loss

- How would you make the computer understand this?