Procedure Overloading

Overloading

- This means that different versions of a procedure can be written to take different types of input, and can be referenced by the same name.
- All Fortran90+ intrinsic functions that take reals are overloaded to take at least both single and double precisions, and sometimes complex as well.
- You can make your own procedures behave similarly.

Fortran Syntax

- Overloaded functions must be defined in a module. The MODULE PROCEDURE structure allows the different procedures to be grouped into a generic name.
- MODULE PROCEDURE is used within an explicit interface. This is the only time a procedure in a module needs or should have an explicit interface.

Fortran Example

```
MODULE triangle
PUBLIC   diagonal
PRIVATE fdiag, ddiag
INTERFACE diagonal
   MODULE PROCEDURE fdiag, ddiag

END INTERFACE

CONTAINS
    REAL FUNCTION fdiag(x)

        REAL x

    END FUNCTION fdiag

    DOUBLE PRECISION FUNCTION ddiag(x)

        DOUBLE PRECISION x

    END FUNCTION ddiag

END MODULE triangle
```

Python “Duck Typing”

- “Duck typing” means that typing is determined by context (“if it walks like a duck…” etc.)
- The simplest way to overload in Python is to use the isinstance() intrinsic.

Python Example

```
def __init__ (self, filename):
    if isinstance (filename,
basestring):
        # filename is a string
    else:
        # try to convert to a list
        self.path = list (filename)
```

Operator Overloading-Fortran

- Fortran can also overload arithmetic operators using the INTERFACE OPERATOR syntax.

```
INTERFACE OPERATOR(+)
    MODULE PROCEDURE addtype
END INTERFACE
```

- This can be useful in dealing with operators on defined types.
- You must write addtype for your type.

Operator Overloading-Python

- Python actually implements basic operations as functions

__add__, __sub__,__mul__,__div__

- You can write these methods for your class
- You can also override comparison operators

__lt__,__le__,__gt__,__ge__,__eq__,__ne__