SNS COLLEGE OF ENGINEERING
Kurumbapalayam (Po), Coimbatore – 641 107

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING

- **A readable, dynamic, pleasant, flexible, fast and powerful language**

Objective

Files and exception: text files, reading and writing files, format operator; command line arguments, errors and exceptions, handling exceptions, modules, packages; Illustrative programs: word count, copy file, Voter's age validation, Marks range validation (0-100).

RECAP

Modules in Python are simply Python files with a .py extension.

- Executing Modules
- Standard Modules
- dir() Function

Packages

- "dotted module names" (ex. *a.b)*
  - Submodule *b* in package *a*
- Saves authors of multi-module packages from worrying about each other's module names
- Python searches through *sys.path* directories for the package subdirectory
- Users of the package can import individual modules from the package
- Ways to import submodules
  - *import sound.effects.echo*
  - *from sound.effects import echo*
- Submodules must be referenced by full name
- An *ImportError* exception is raised when the package cannot be found

Importing * From a Package

- * does not import all submodules from a package
- Ensures that the package has been imported, only importing the names of the submodules defined in the package
- *import sound.effects.echo*

  import sound.effects.surround

  *from sound.effects import **

Intra-package References

- Submodules can refer to each other
  - *Surround* might use *echo* module
  - *import echo* also loads *surround* module
- *import* statement first looks in the containing package before looking in the standard module search path
- Absolute imports refer to submodules of sibling packages
  - *sound.filters.vocoder* uses *echo* module

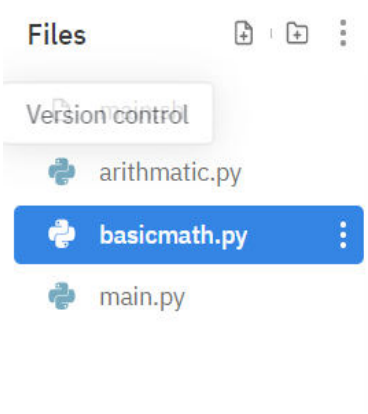
    from sound.effects import echo
- Can write explicit relative imports
  - *from . import echo*
  - *from .. import formats*
  - *from ..filters import equalizer*

Packages in Multiple Directories

- *_path_* is a list containing the name of the directory holding the package's *_init_.py*
- Changing this variable can affect fututе searches for modules and subpackages in the package
- Can be used to extend the set of modules in a package
- Not often needed

Usage

Main.py using the math module



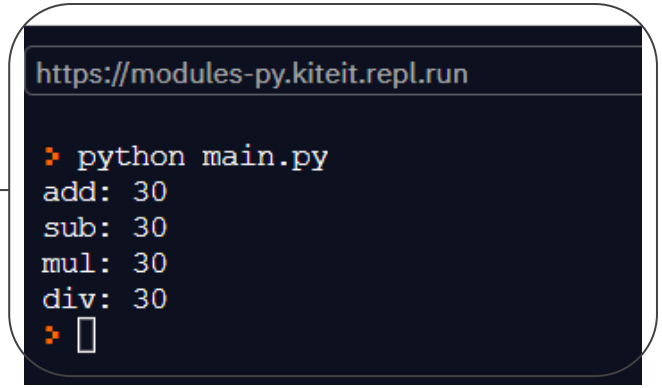
basicmath.py

```
1  def add(a,b):
2      return a+b
3  def sub(a,b):
4      return a+b
5  def mul(a,b):
6      return a+b
7  def div(a,b):
8      return a+b
```

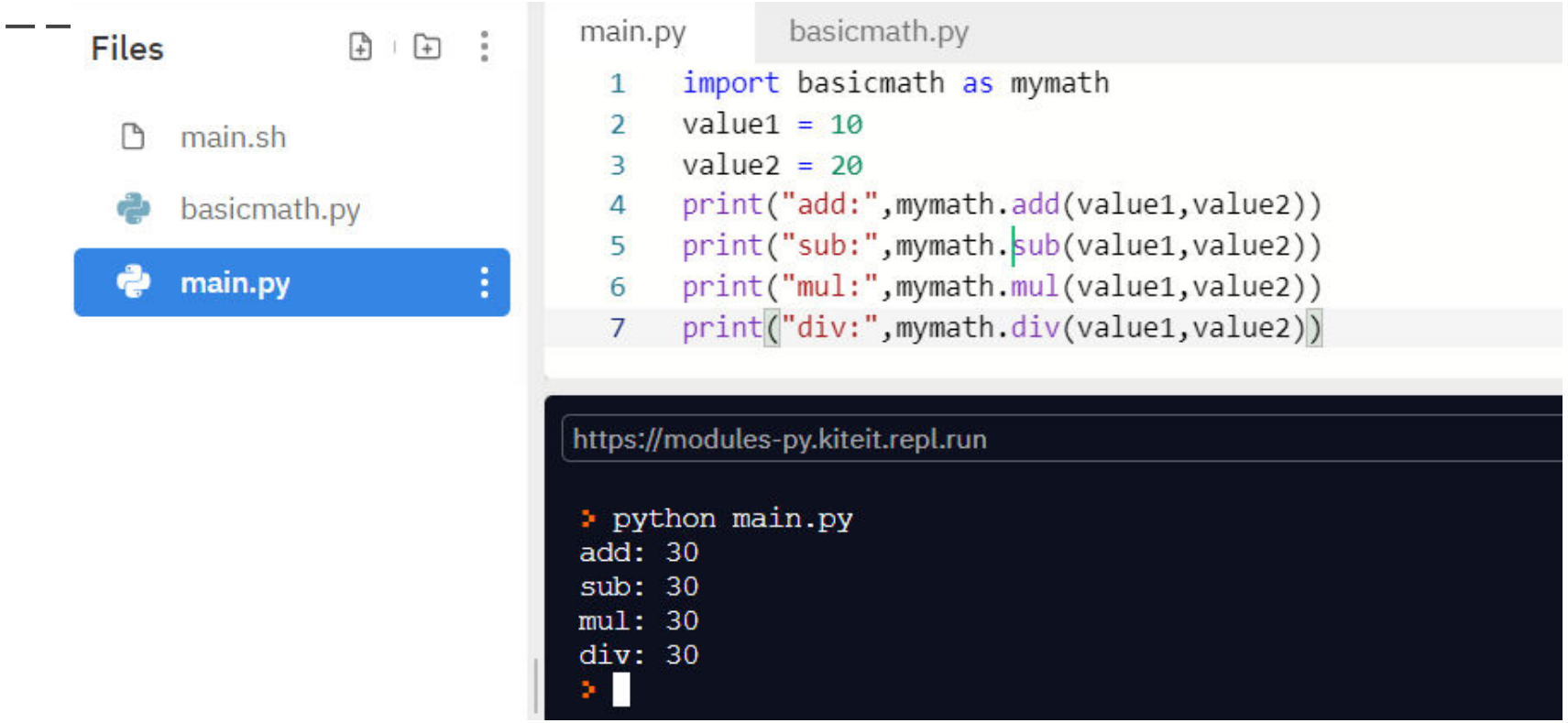
main.py

```
1  import basicmath
2  value1 = 10
3  value2 = 20
4  print("add:",basicmath.add(value1,value2))
5  print("sub:",basicmath.sub(value1,value2))
6  print("mul:",basicmath.mul(value1,value2))
7  print("div:",basicmath.div(value1,value2))
```

Basicmath.py (module contains useful arithmetic functions)

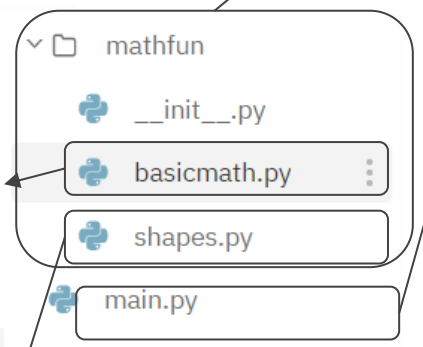


Package name (To avoid conflicts)



Packages

Math Fun package (folder with more modules and __init__.py)



mathfun/__init__.py

```
1  def add(a,b):
2      return a+b
3  def sub(a,b):
4      return a+b
5  def mul(a,b):
6      return a+b
7  def div(a,b):
8      return a+b
```

mathfun/__init__.py | main.py

```
1  from mathfun import basicmath
2  from mathfun import shapes
3  value1 = 10
4  value2 = 20
5  print("add:",basicmath.add(value1,value2))
6  print("sub:",basicmath.sub(value1,value2))
7  print("mul:",basicmath.mul(value1,value2))
8  print("div:",basicmath.div(value1,value2))
9  print("area_of_rectangle:",shapes.areaOfRectangle(value1,value2))
10 print("area_of_tritangle:",shapes.areaOfTriangle(value1,value2))
```

mathfun/__init__.py | mathfun/shapes.py

```
1  def areaOfRectangle(length, width):
2      return length*width
3
4  def areaOfTriangle(base, height):
5      return 0.5*base*height
```

https://modules-py.kiteit.repl.run

```
> python main.py
add: 30
sub: 30
mul: 30
div: 30
area_of_rectangle: 200
area_of_tritangle: 100.0
>
```

SUMMARY

Packages are a way of structuring Python's module namespace by using "dotted module names". For example, the module name A.B designates a submodule named B in a package named A. Just like the use of modules saves the authors of different modules from having to worry about each other's global variable names, the use of dotted module names saves the authors of multi-module packages like NumPy or Pillow from having to worry about each other's module names.


Thank
you