

# Functional Programming with Python

λ



$\lambda f.(\lambda x.(f(x x))\lambda x.(f(x x)))$

```
def grep(filename, regexp):  
    """ return list of lines matching regexp from file """  
    ret = []  
    for line in open(filename):  
        if re.compile(regexp).search(line):  
            ret.append(line.rstrip())  
    return ret
```

```
def list_large_files(directory, max_size):
    """ return list of files larger than max_size """
    ret = []
    for file in os.listdir(directory):
        if os.stat(directory + file)[6] > max_size:
            ret.append(directory+file)
    return ret
```

```
def grep(filename, regexp):
    """ return list of lines matching regexp from file """
    ret = []
    for line in open(filename):
        if re.compile(regexp).search(line):
            ret.append(line.rstrip())
    return ret
```

```
def list_large_files(directory, max_size):  
    """ return list of files larger than max_size """  
    ret = []  
    for file in os.listdir(directory):  
        if os.stat(directory + file)[6] > max_size:  
            ret.append(directory+file)
```

# GENERATOR

```
def grep(filename, regexp):  
    """ return list of lines matching regexp from file """  
    ret = []  
    for line in open(filename):  
        if re.compile(regexp).search(line):  
            ret.append(line.rstrip())  
    return ret
```

```
def list_large_files(directory, max_size):
    """ return list of files larger than max_size """
    ret = []
    for file in os.listdir(directory):
        if os.stat(directory + file)[6] > max_size:
            ret.append(directory+file)
    return ret
```

# FILTER

```
def grep(filename, regexp):
    """ return list of lines matching regexp from file """
    ret = []
    for line in open(filename):
        if re.compile(regexp).search(line):
            ret.append(line.rstrip())
    return ret
```

```
def list_large_files(directory, max_size):
    """ return list of files larger than max_size """
    ret = []
    for file in os.listdir(directory):
        if os.stat(directory + file)[6] > max_size:
            ret.append(directory+file)
    return ret
```

# MAP

```
def grep(filename, regexp):
    """ return list of lines matching regexp from file """
    ret = []
    for line in open(filename):
        if re.compile(regexp).search(line):
            ret.append(line.rstrip())
    return ret
```

```
def list_large_files(directory, max_size):
    """ return list of files larger than max_size """
    ret = []
    for file in os.listdir(directory):
        if os.stat(directory + file)[6] > max_size:
            ret.append(directory+file)
    return ret
```

# REDUCE

```
def grep(filename, regexp):
    """ return list of lines matching regexp from file """
    ret = []
    for line in open(filename):
        if re.compile(regexp).search(line):
            ret.append(line.rstrip())
    return ret
```

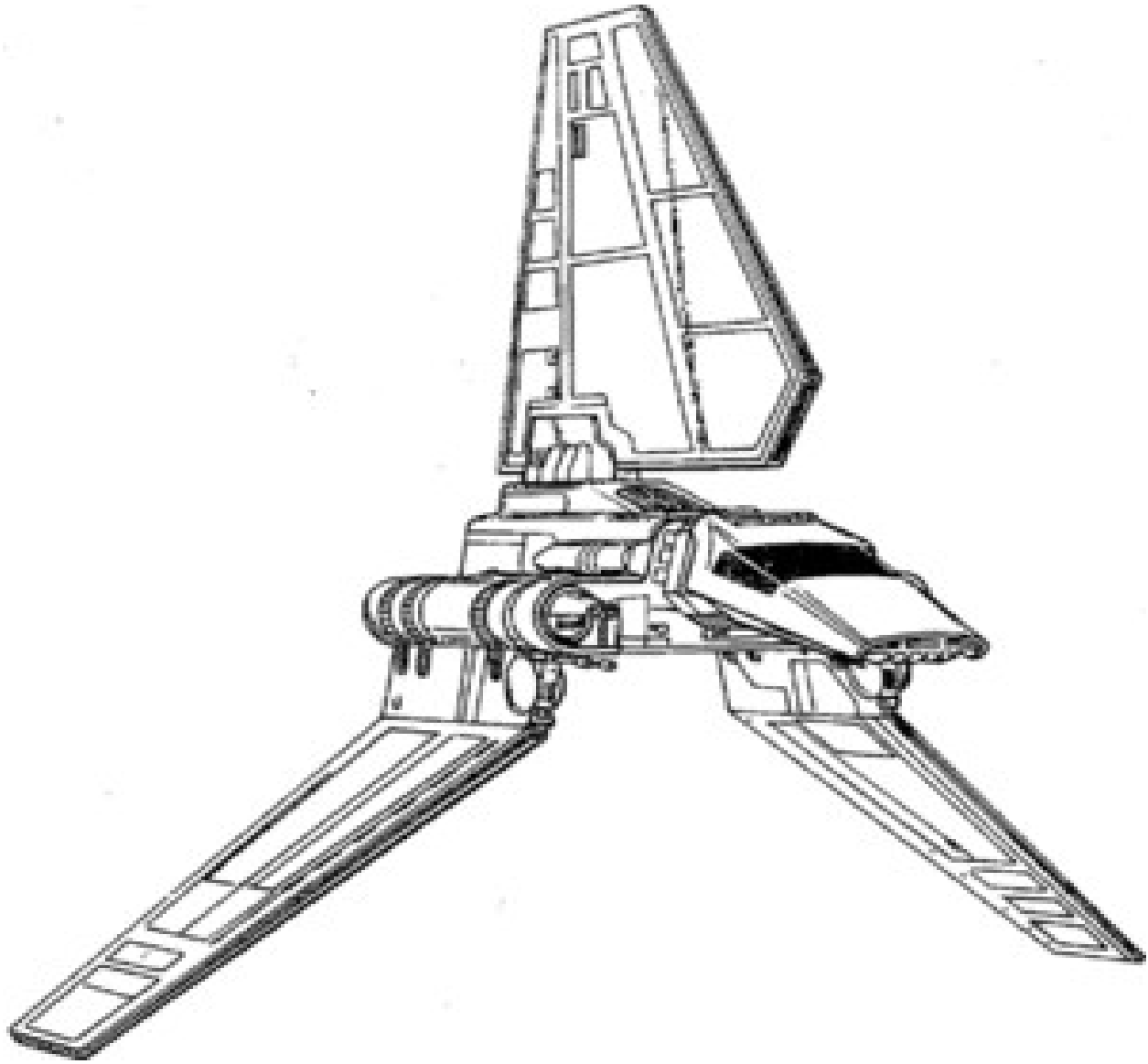
```
def generic(filtrate, generate):  
    """ generically handle filtering """  
    for i in generate():  
        if filtrate(i):  
            yield i
```

`itertools.ifilter(predicate, iterable)`

```
itertools.ifilter(  
    re.compile(regex).search,  
    open(filename)  
)
```

```
itertools.filter(  
    lambda(x): os.stat(x)[6] > 2TB,  
    os.listdir("."))  
)
```





```
def say(*args):  
    print ' '.join( map( lambda(x):x.rstrip(), args ) )
```

```
imap(say,  
    ifilter(  
        lambda(x): os.stat(x)[6] > 2TB,  
        imap(  
            lambda(x): "/etc/" + x,  
            os.listdir(".")  
        )  
    )  
)
```

```
def say(*args):  
    print ' '.join( map( lambda(x):x.rstrip(), args ) )
```

```
imap(say,  
    ifilter(  
        lambda(x): os.stat(x)[6] > 2TB,  
        imap(  
            lambda(x): "/etc/" + x,  
            os.listdir(".")  
        )  
    )  
)
```



Generate

Map

Filter

Reduce

Generate

Map

Filter

Reduce