PYTHON – STATEMENTS

Statement	Role	Example
Assignment	Creating references	`a, *b = 'good', 'bad', 'ugly'`
Calls and other expressions	Running functions	`log.write("spam, ham")`
`print` calls	Printing objects	`print('The Killer', joke)`

`if/elif/else`	Selecting actions	`if "python" in text:`
		`    print(text)`

`for/else`	Sequence iteration	`for x in mylist:`
		`    print(x)`

`while/else`	General loops	`while X > Y:`
		`    print('hello')`

`pass`	Empty placeholder	`while True:`
		`    pass`

`break`	Loop exit	`while True:`
		`    if exittest(): break`

`continue`	Loop continue	`while True:`
		`    if skiptest(): continue`

FOR ELSE

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print n, 'equals', x, '*', n/x
            break
    else:
    # loop fell through without finding a factor
        print n, 'is a prime number'
```

FOR ELSE

```
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

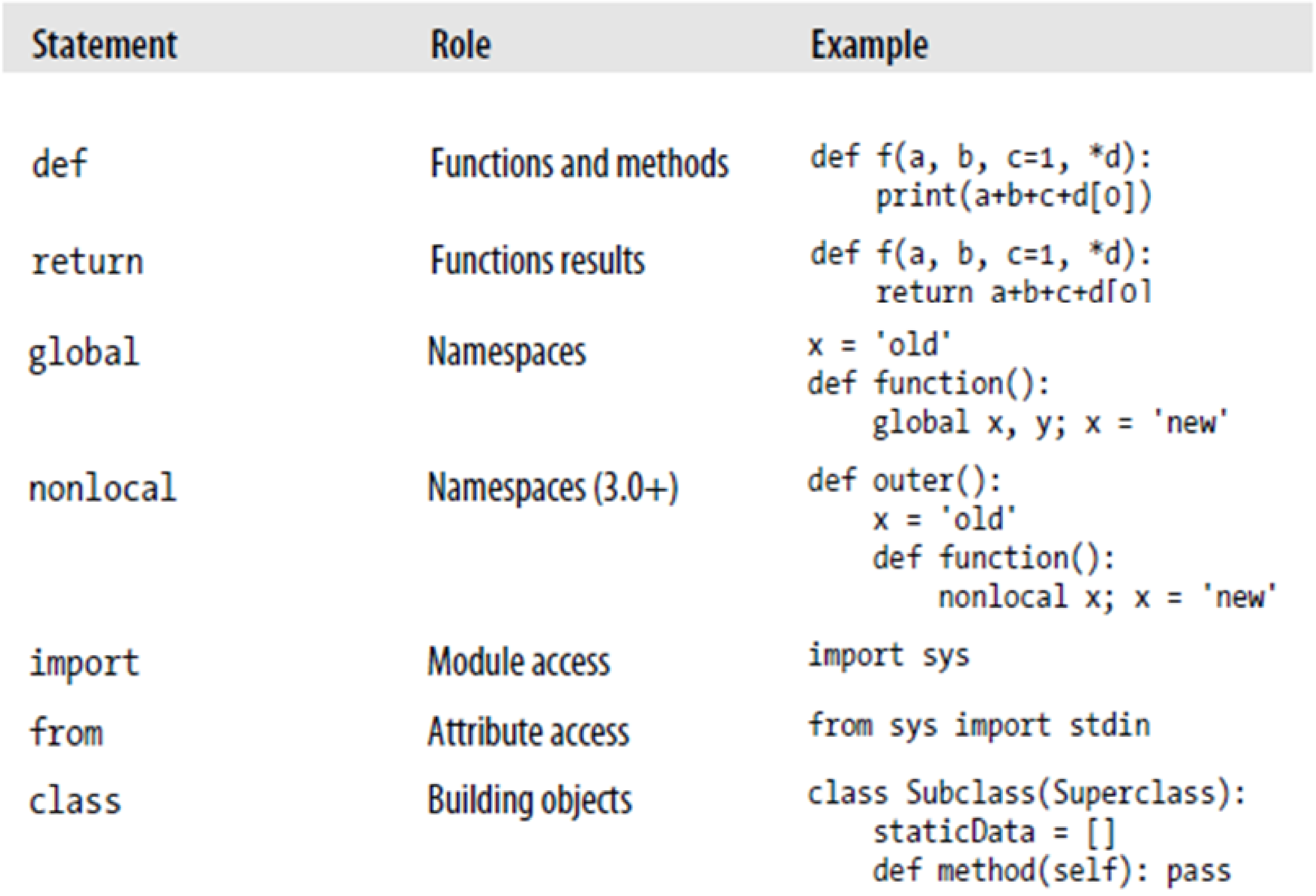
PASS

```
class MyClass(object):
    def meth_a(self):
        pass

    def meth_b(self):
        print "I'm meth_b"
```

WHILE ELSE

```
n = 5
while n != 0:
    print n
    n -= 1
else:
    print "what the..."
```



Statement	Role	Example

`def`	Functions and methods	`def f(a, b, c=1, *d):`
		`    print(a+b+c+d[0])`

`return`	Functions results	`def f(a, b, c=1, *d):`
		`    return a+b+c+d[0]`

`global`	Namespaces	`x = 'old'`
		`def function():`
		`    global x, y; x = 'new'`

`nonlocal`	Namespaces (3.0+)	`def outer():`
		`    x = 'old'`
		`    def function():`
		`        nonlocal x; x = 'new'`

`import`	Module access	`import sys`
`from`	Attribute access	`from sys import stdin`

`class`	Building objects	`class Subclass(Superclass):`
		`    staticData = []`
		`    def method(self): pass`

NONLOCAL

```
>>> def outside():
        msg = "Outside!"
        def inside():
            msg = "Inside!"
            print(msg)
        inside()
        print(msg)

>>> outside()
Inside!
Outside!
```

NONLOCAL

```
>>> def outside():
        msg = "Outside!"
        def inside():
            nonlocal msg
            msg = "Inside!"
            print(msg)
        inside()
        print(msg)

>>> outside()
Inside!
Inside!
```

NONLOCAL

```
>>> def outside():
        d = {"outside": 1}
        def inside():
            d["inside"] = 2
            print(d)
        inside()
        print(d)

>>> outside()
{'inside': 2, 'outside': 1}
{'inside': 2, 'outside': 1}
```

Statement	Role	Example

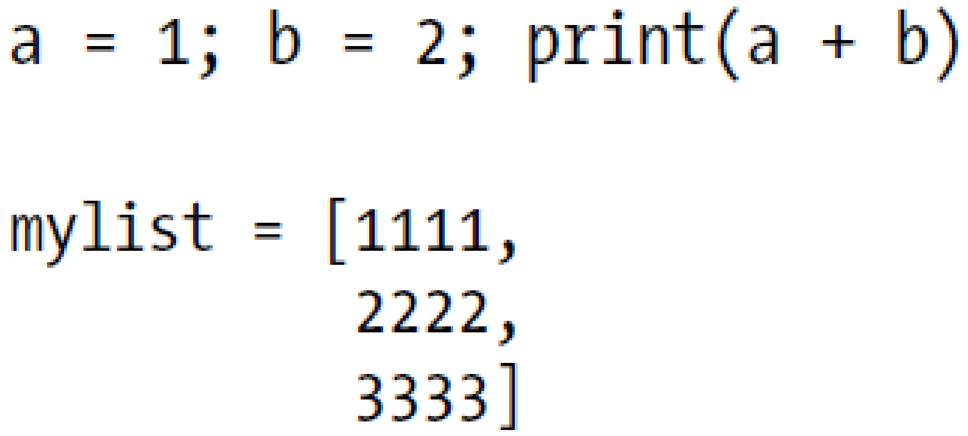
`try/except/ finally`	Catching exceptions	`try:`
		`    action()`
		`except:`
		`    print('action error')`

`raise`	Triggering exceptions	`raise EndSearch(location)`
`assert`	Debugging checks	`assert X > Y, 'X too small'`

`with/as`	Context managers (2.6+)	`with open('data') as myfile:`
		`    process(myfile)`

`del`	Deleting references	`del data[k]`
		`del data[i:j]`
		`del obj.attr`
		`del variable`

Multiple statements per line or lines per statement

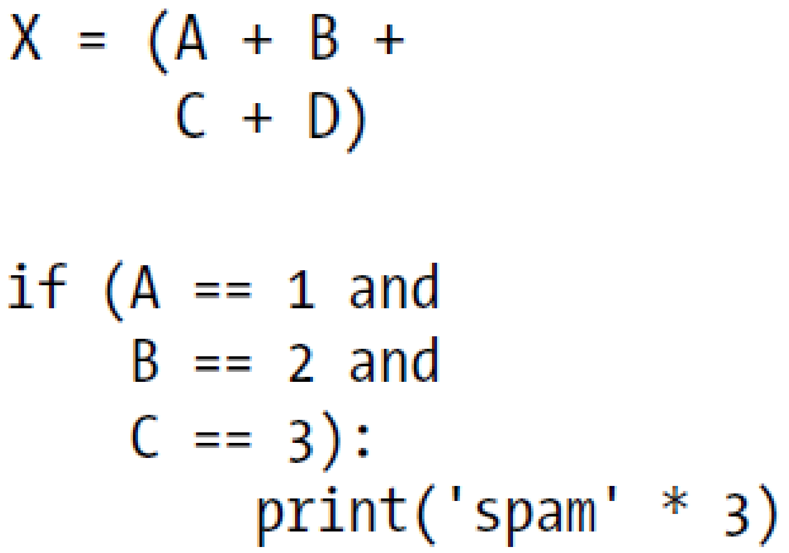


```
a = 1; b = 2; print(a + b)

mylist = [1111,
          2222,
          3333]
```

Parentheses

Parentheses are the catchall device—because any expression can be wrapped up in them, simply inserting a left parenthesis allows you to drop down to the next line and continue your statement:



```
X = (A + B +
     C + D)

if (A == 1 and
    B == 2 and
    C == 3):
        print('spam' * 3)
```

input

All “input” is formatted to a string by default..

What can go wrong with this code?

```
while True:
      reply = input('Enter text:')
      if reply == 'stop': break
      print(int(reply) ** 2)
      print(‘Go Again \n')
```

Test input

```
while True:
 reply = input('Enter text:  ')
 if reply == 'stop':
     break
 elif  not  reply.isdigit():
     print('Bad!' * 8)
 else:
     print(int(reply) ** 2)
     print(‘Next.. \n')
```

Handling errors with a try

```
while True:
 reply = input('Enter text:')
 if reply == 'stop': break
 try:
     num = int(reply)
 except:
     print('Bad!' * 8)
 else:
     print(int(reply) ** 2)
     print('Bye')
```

Nesting

```
while True:
 reply = input('Enter text:')
 if reply == 'stop':
        break
 elif  not reply.isdigit():
        print('Bad!' * 8)
 else:
        num = int(reply)
        if num < 20:
                print('low')
        else:
                print(num ** 2)
                print('Bye')
```

In Class Activity

Write a program that takes a list of objects and proceeds through the list; iterating each iteratable object and simply stating the type of each non iterable object.