ME1 Computing

The programmer got stuck in the shower because the instructions on the shampoo bottle said ...

Lather, Rinse, Repeat

Provide feedback (anonymously) at:

www.menti.com

with code **63 53 77**

Building complex algorithms

Top-down algorithm: outsourcing problems

Call the carpenter

Call the builder

Call the plumber

Call the electrician

Top-down algorithm: outsourcing problems

Call the carpenter

Call the electrician

Call the plumber

Modular programming: functions

Each person is specialised / expert in a specific function

Similarly, in Computing, we can have specialised scripts to performs specific functions

Specialised scripts, that can be called by other programmers too, are called **functions**

Each function has a name

```
def a():
```

a is a function

```
a = 2
print(a)
```

a is a variable

```
def carpenter():

def plumber():
```

Defining a function

```
def name():
```

```
def carpenter():
```

```
def carpenter():
    # all the instructions written in here belong to this func only
    ToDo = ['lamp',wood','door']
    for item in ToDo
        print('Fix'+item)
```

```
def builder():
    # all the instructions written in here belong to this func only
    ToDo = ['brick','sand','water']
    for item in ToDo
        print('Add'+item)
```

```
def plumber():
    # all the instructions written in here belong to this func only
    tap = 'broken'
    if tap == 'broken':
        print('Repair tap')
    cost = 5
```

Invoking a function

We need to call it from the main program

```
def carpenter():
    ToDo = ['lamp','wood','door']
    for item in ToDo
        print('Fix'+item)
```

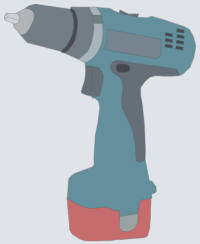
```
def builder():
    ToDo = ['brick','sand','water']
    for item in ToDo
        print('Add'+item)
```

Main script or caller

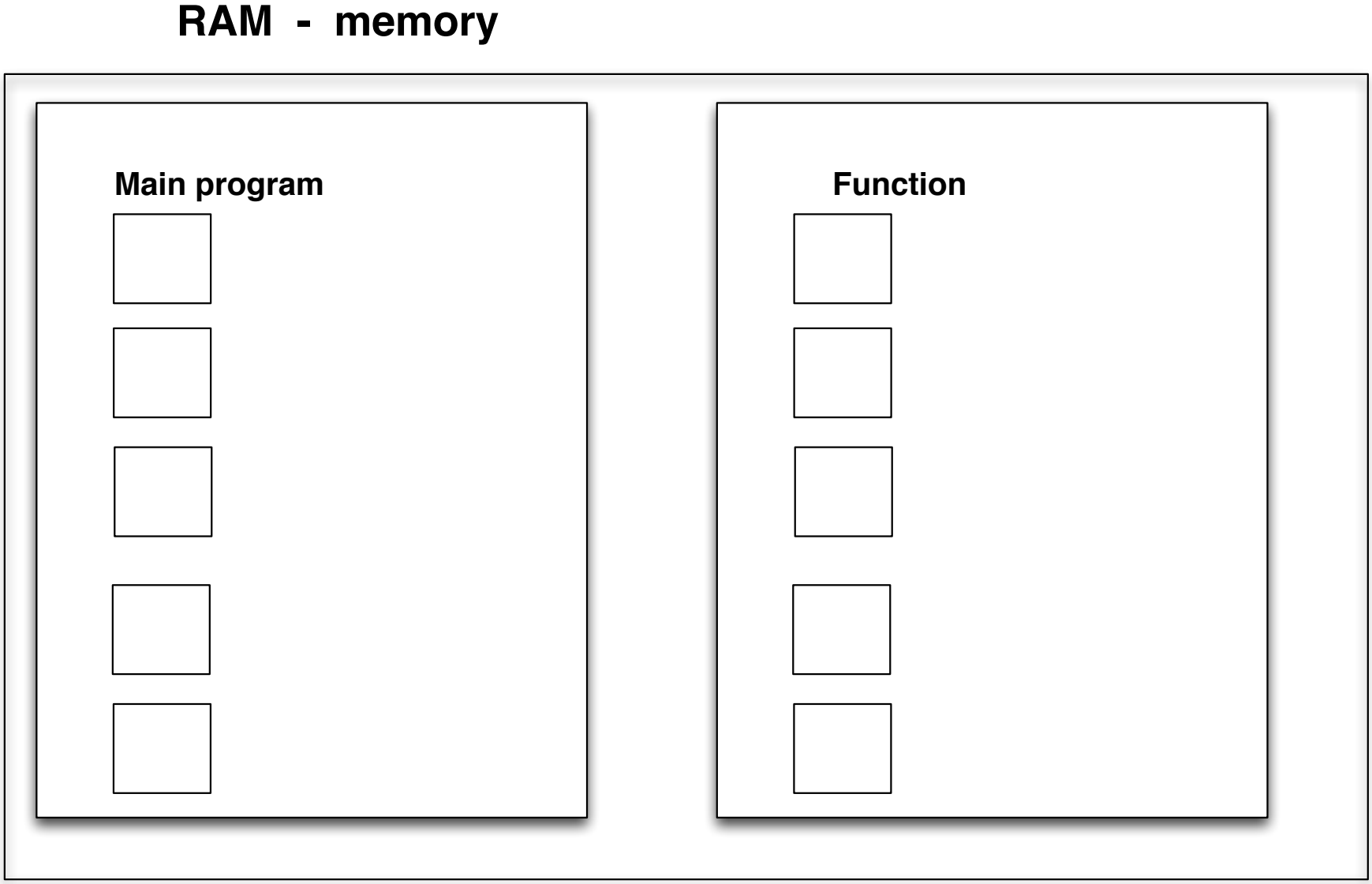
```
carpenter()

builder()
```

Scope of variables



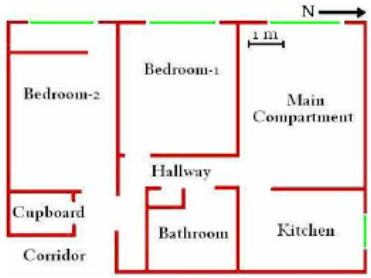
How do variables and arguments work?

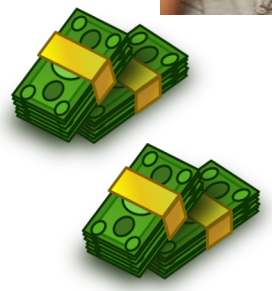


Input arguments

```
def carpenter(money,map):
```

arguments



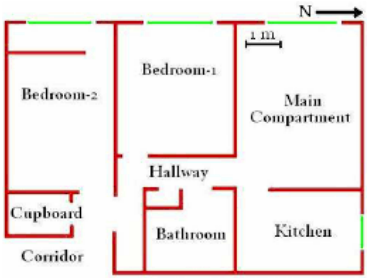


Need money and map

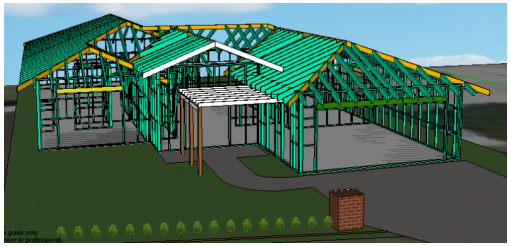
Need money and map

Output arguments

```
def carpenter(money,map):
    timber = wood + money
    layout = map + pen
    house = timber + layout
    return house
```

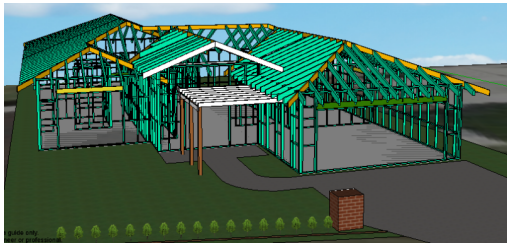


Need money and map



Output arguments

```
def carpenter(money,map):
    timber = wood + money
    layout = map + pen
    house = timber + layout
    return house
```



```
money = 5000
map = 'map.jpg'
outcome = carpenter(money,map)
```

Multiple output arguments (tuples)

```
def SumAndPower(a,b):
    Sum = a + b
    Power = a ** b
    return (Sum,Power)



a = 2
b = 4
(s,p) = SumAndPower(a,b)
print(s,p)
```

No output arguments

```
def DesperateCall(name):
    print('Dear')
    print(name)
    print('I Love you.')
    print('Fancy a date tonight?')
    print('XXX')
```

How do variables and arguments work?

```
def Add10(n):
    new = n + 1000
    return new
```

Case 1:
as we know until now

```
% Main program
a = input('My salary')
b = Add10(int(a))
print(b)
```

Case 2:
returning variable same
as argument

```
% Main program
a = input('My salary')
a = AddBonus(int(a))
print(a)
```

How do variables and arguments work?

```
def Add10(n):
    n = n + 1000
    return n
```

Case 3:
output variable same as input

```
% Main program
a = input('My salary')
a = AddBonus(int(a))
a = a * 2
print(a)
```

How do variables and arguments work?

```
def MyFun(a):
    a = a * 2
    res = a
    print(a)
    return res
```

Case 4:
Changes to the input variables do not reflect back in the main program

```
% Main program
a = input('Gimme a number');
b = MyFun(int(a));
print(b)
print(a)
```

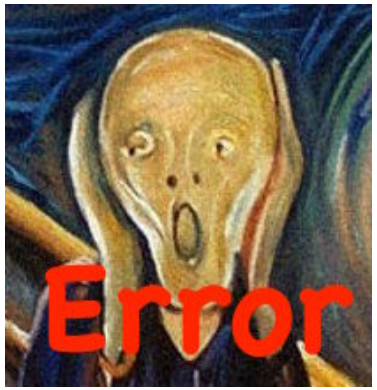
This case is cause of common pitfalls!

How do variables and arguments work?

```
def SumAndPower(a,b):
    Sum = a + b
    Power = a ** b
    return (Sum,Power)
```

```
a = 2
b = 4
(s,p) = SumAndPower(a)
print(s,p)
```

Case 5:
not passing enough arguments

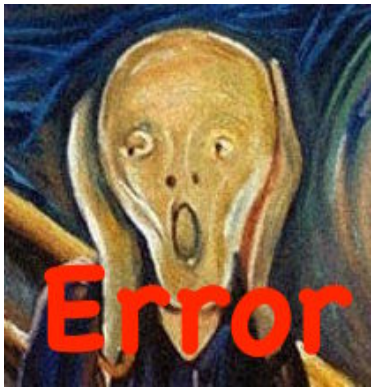


How do variables and arguments work?

```
def SumAndPower(a,b):
    Sum = a + b
    Power = a ** b
    return (Sum,Power)
```

```
a = 2
b = 4
(s,p) = SumAndPower(b,a)
print(s,p)
```

Case 6:
Arguments in wrong order

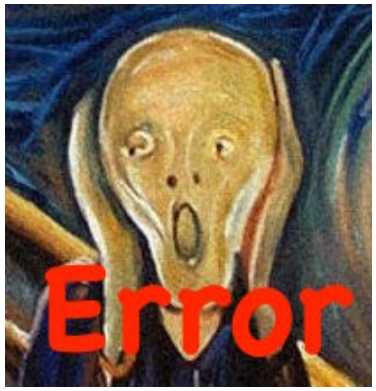


How do variables and arguments work?

```
def Add(d,e)
    c = d + e
   return
```

Case 7
Not assigning a return variable

```
% Main program
a = 4
b = input('Gimme a number');
a = Add(a,int(b));
print(a)
```



Arguments can be lists too

```
a = [1,2,3,4,5,6,7,8,9,10]
```

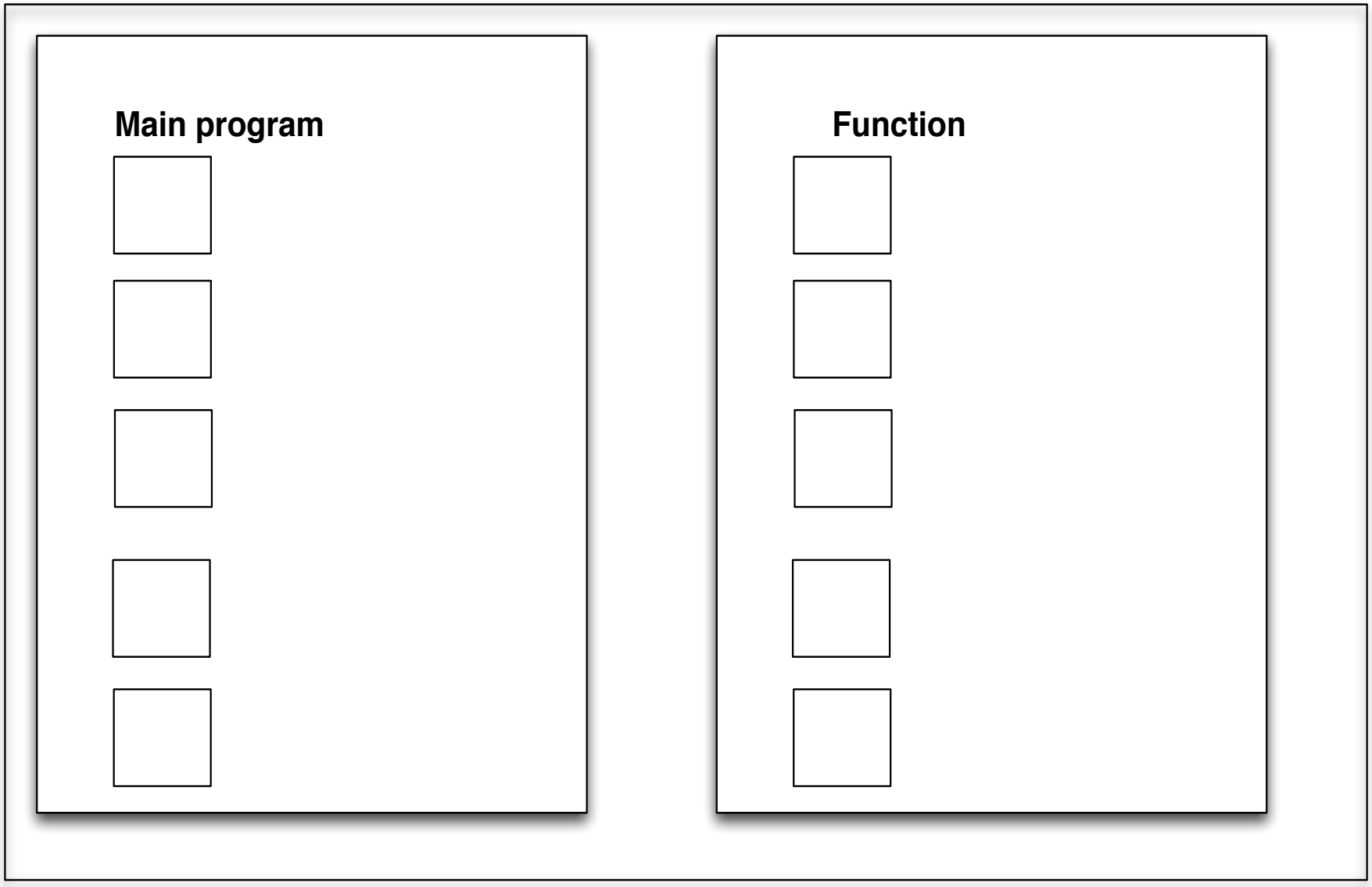
```
def square(x):
    y = []
    for i in range(0,len(x))
        y = y + [x(i)**2];
    return y
```

```
a = [1,2,3,4,5,6,7,8,9,10]

a = [1,2,3]

a = [1]
```

RAM - memory



How do variables and arguments work?