Python

Functions

Function

What is it

Why good for programming

Other names:

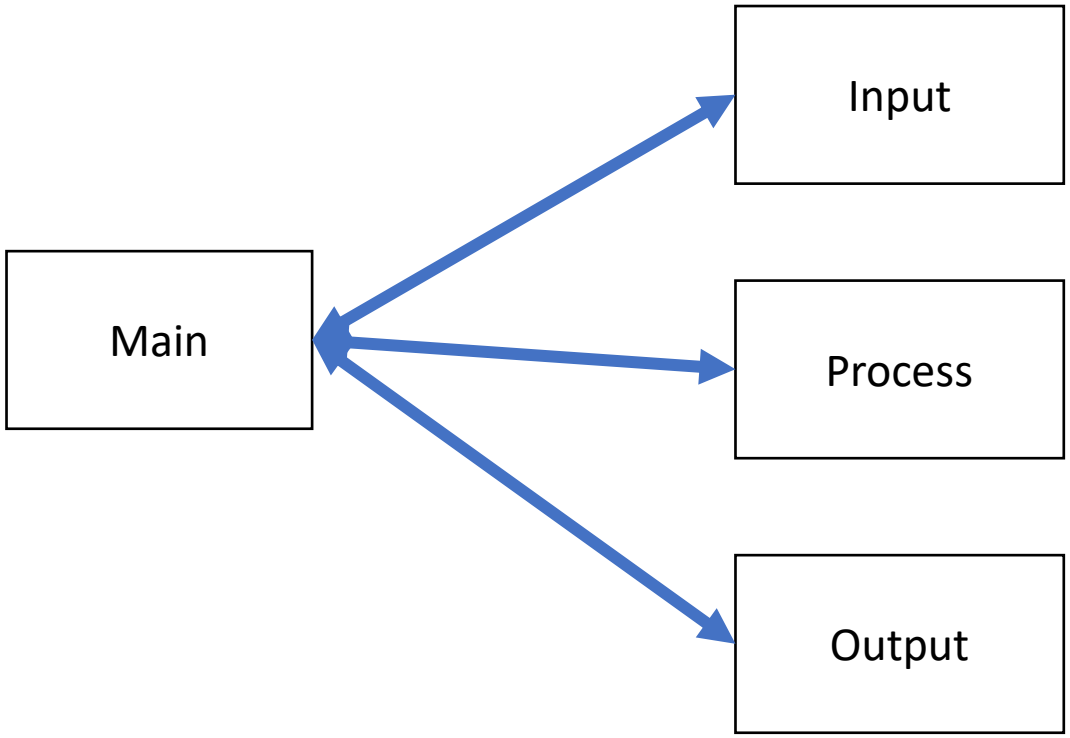
- subroutines
- modules
- Structured programming

Functions

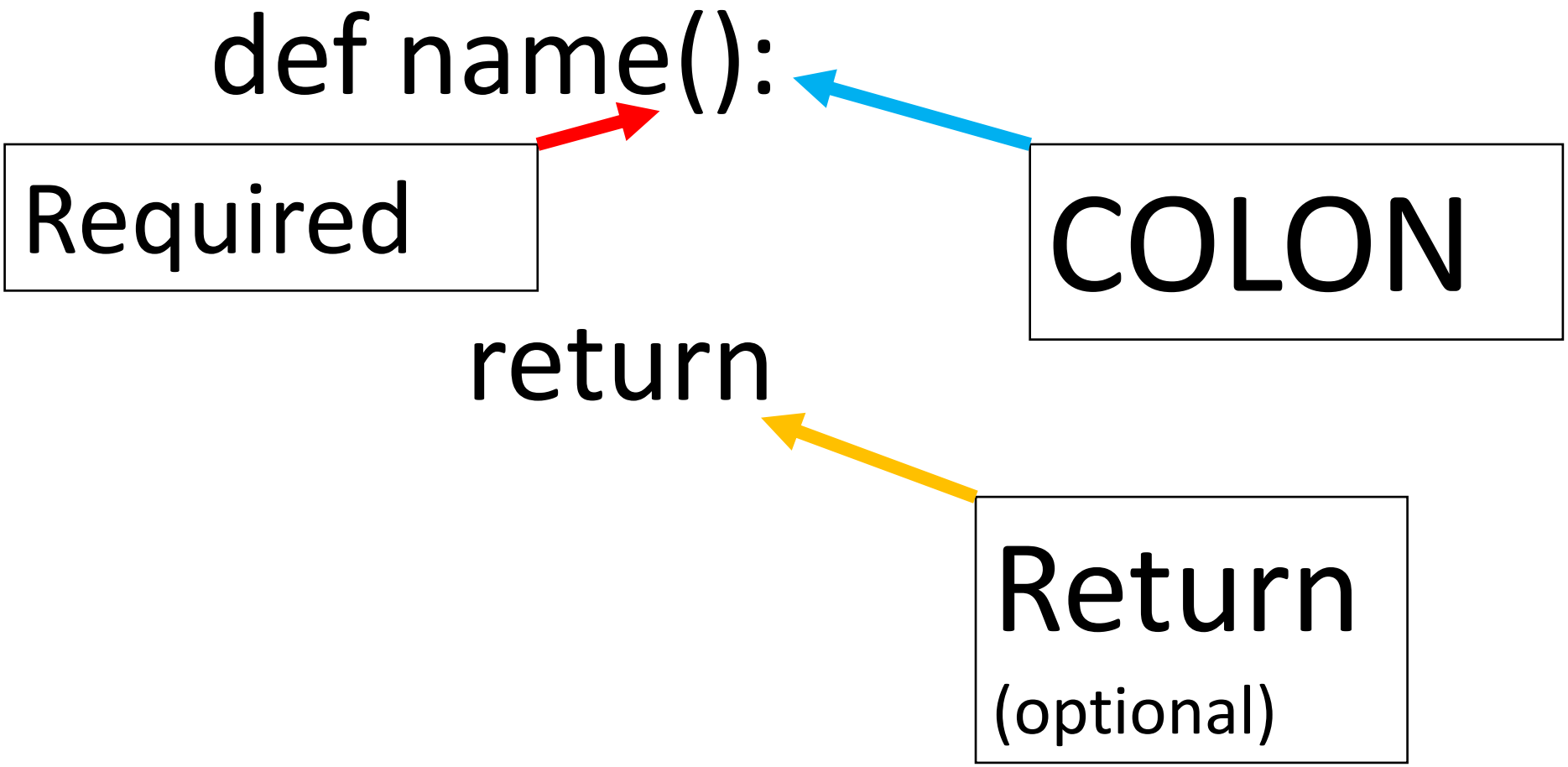
A small set of code that looks like a mini program.

Functions can be called and can have data passed to the function and send data back to the calling statement.

Functions

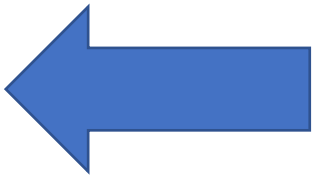


Python - def



Statements

```
def name():

    return
```



Statements go here

Statements must be indented (4 spaces)

function Naming Rules

1. Characters allowed
   a) A-Z, a-z
   b) Numbers 0-9
   c) Special: underscore
2. Case sensitive
3. Cannot be reserve word
4. NO embedded spaces

function Naming Rules - more

5. Must start with an alpha chr
6. Name must be unique
7. Cannot be a Module name
8. Cannot be a program name

Naming Rules more:

- Length: any (be reasonable)
- Readability is very important.
- Descriptive names are very useful.
- Avoid using the lowercase letter ‘l’, uppercase ‘O’, and uppercase ‘I’.

Same as variable naming

Using a Function

When using a function
This is known as
Calling the function

The shell Program

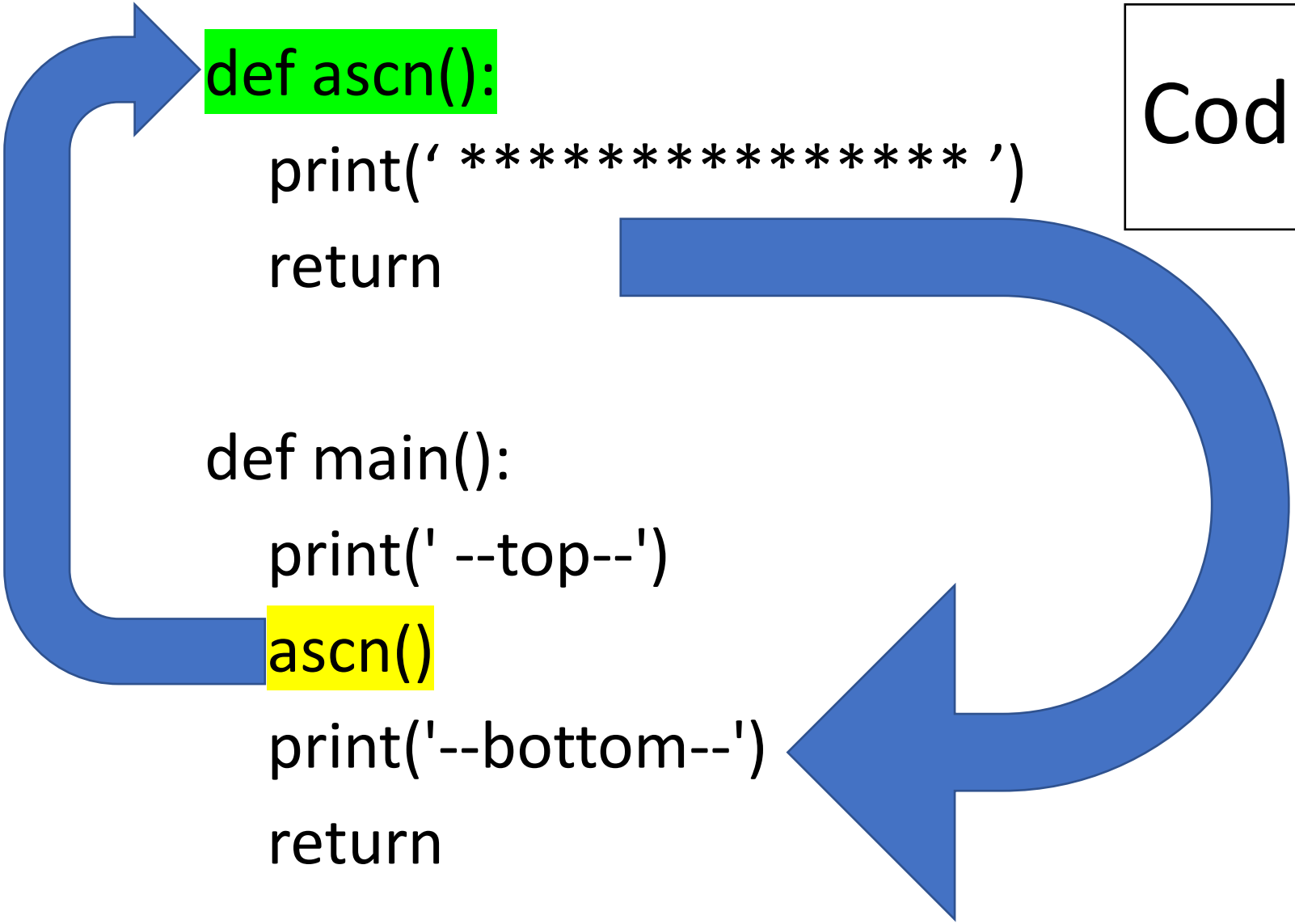
```
def main():

    return

main()
```

The Function

The Call to the function

Code

```
def ascn():
    print(‘ ********** ’)
    return

def main():
    print(' --top--')
    ascn()
    print('--bottom--')
    return

main()
```



```
def ascn():
    print(‘ ********** ’)
    return

def main():
    print(' --top--')
    ascn()
    print('--bottom--')
    return

main()
```

Try this

Simplify program

Put repetitive code into function
Call the function when needed

Also, one time code into a function.

GOAL: to simplify the main line.

Example

before

```
def main():
    print(' ********** ')
    print(' *        * ')
    print(' ********** ')
    print(' This is a line')
    print(' ********** ')
    print(' *        * ')
    print(' ********** ')
    print(' Line two')
    print(' ********** ')
    print(' *        * ')
    print(' ********** ')
main()
```

after

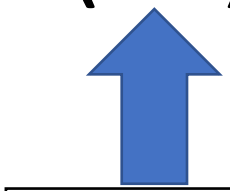
```
def ascn():
    print(' ********** ')
    print(' *        * ')
    print(' ********** ')
    return

def main():
    ascn()
    print(' This is a line')
    ascn()
    print(' Line two')
    ascn()
main()
```

a1,a2

Passing data

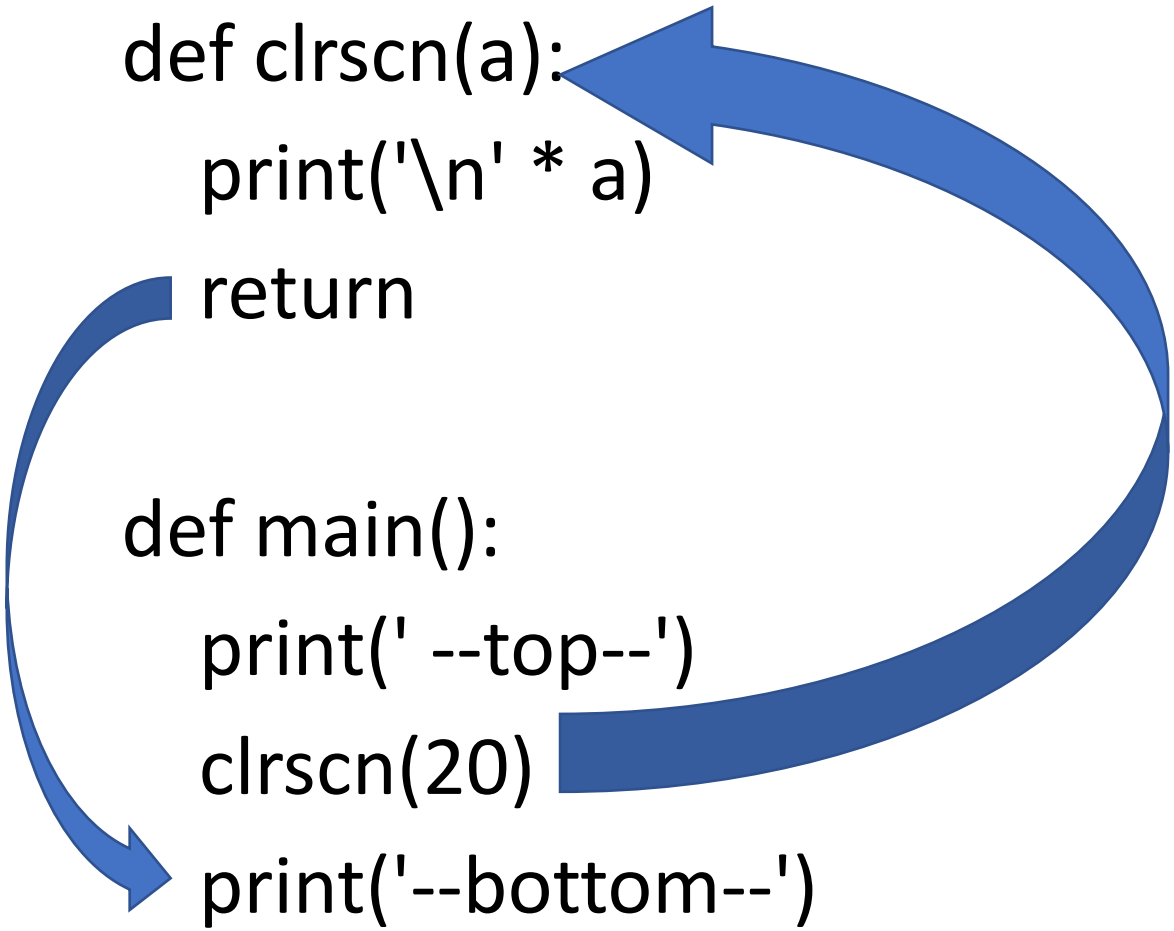
A hard coded value can be passed to a function

Example: countit(20)



Explicit coding

```
def clrscn(a):
    print('\n' * a)
    return

def main():
    print(' --top--')
    clrscn(20)
    print('--bottom--')
    return

main()
```



Type this in

Passing Data

A variable that has a value can be passed to a function

Example: clrscn(xx)

```
def clrscn(a):
    print('\n' * a)
    return

def main():
    xx = 20
    print(' --top--')
    clrscn(xx)
    print('--bottom--')
    return

main()
```

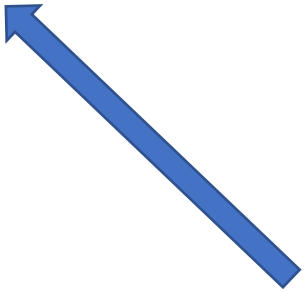
note

```
def clrscn(a):
    print('\n' * a)
    return

def main():
    xx = 20
    print(' --top--')
    clrscn(xx)
    print('--bottom--')
    return

main()
```

Type this in

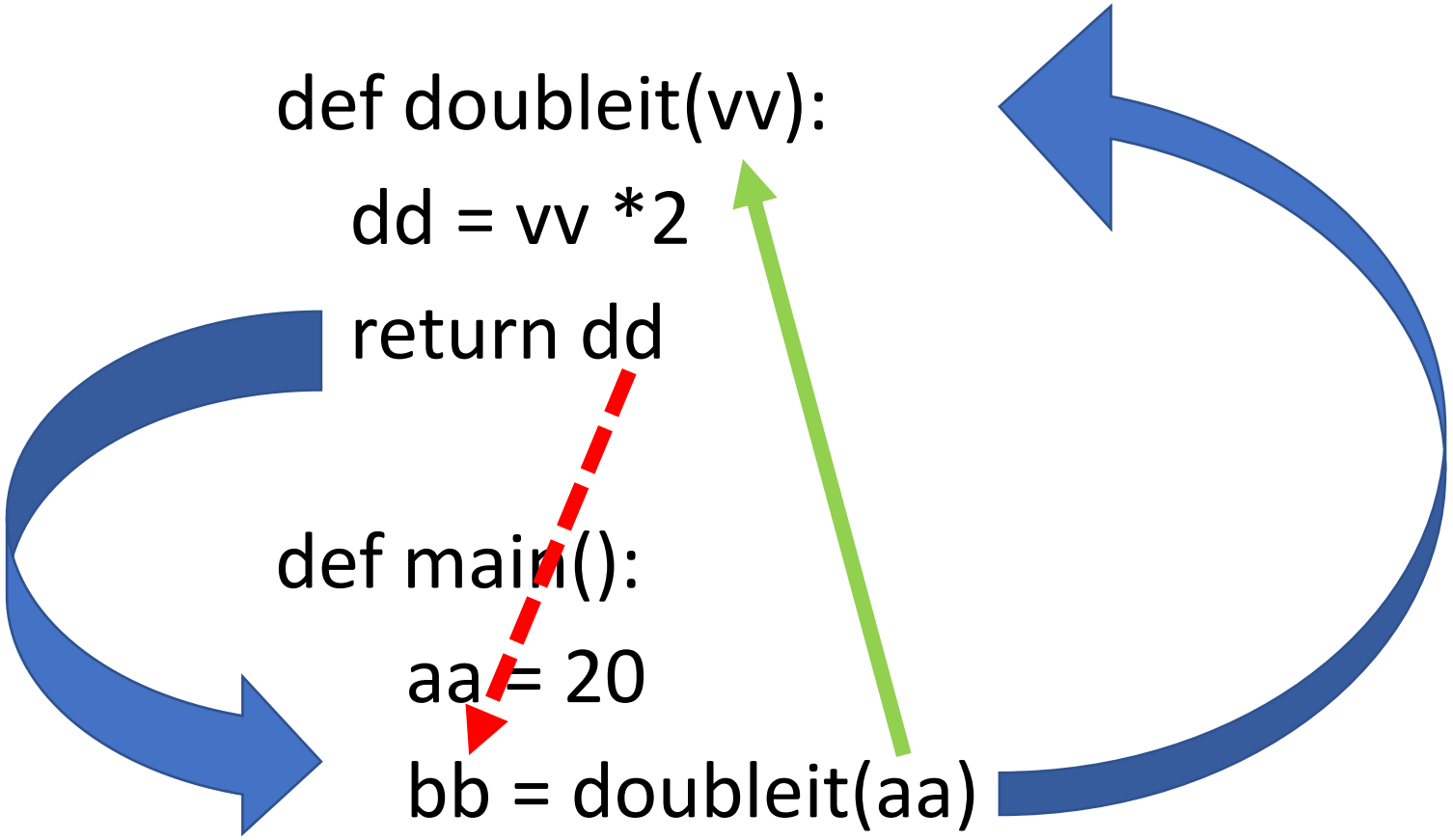
Return value from Function

```
def doubleit(vv):
    dd = vv *2
    return dd
```



Received value in passed variable

Value to be passed back to calling statement



```
def doubleit(vv):
    dd = vv *2
    return dd

def main():
    aa = 20
    bb = doubleit(aa)
    print(‘ double of ‘, aa, ‘ is ‘, bb)
    return

main()
```

```
def doubleit(vv):
    dd = vv *2
    return dd

def main():
        aa = 20
        bb = doubleit(aa)
        print(‘ double of ‘, aa, ‘ is ‘, bb)
        return

main()
```

Try this

Functions can return data only

```
def getdata():
    dd = float(input('enter number -> '))
    return dd

def main():
        bb = getdata()
        print(' Number entered is: ', bb)
        return

main()
```

Functions table

Get	return
None	None
Get	None
None	Return
get	return

Local variables

A variable that is defined and used only inside a function is known as local

Local Variables

Even variable created (defined) in Main are local.

Local variables

Variables in multiple functions can have the same name because they are local.

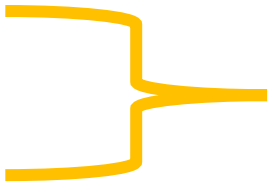
However, the variables with the same name WILL NOT have the same data.

Global variables

A variable can be defined as global, that way, one or more functions can use the variable and the data in the varaiable. The data in the variable can be changed by the function.

When using global, be careful of duplicate names.

```
a=0.0
b=0.0
c=0.0
def getdata():
    a = float(input(" Enter a number; "))
    return a;
def showdata(b):
    print('value is: ',b)
    c = b  *2
    return c
def main():
    b = getdata()
    c = showdata(b)
    print(' c is: ',c)
    return
main()
```



Global Variables

Try this

Built in functions

float()

int()

str()

len()

Placeholder

Functions can be created that do nothing but will be developed later

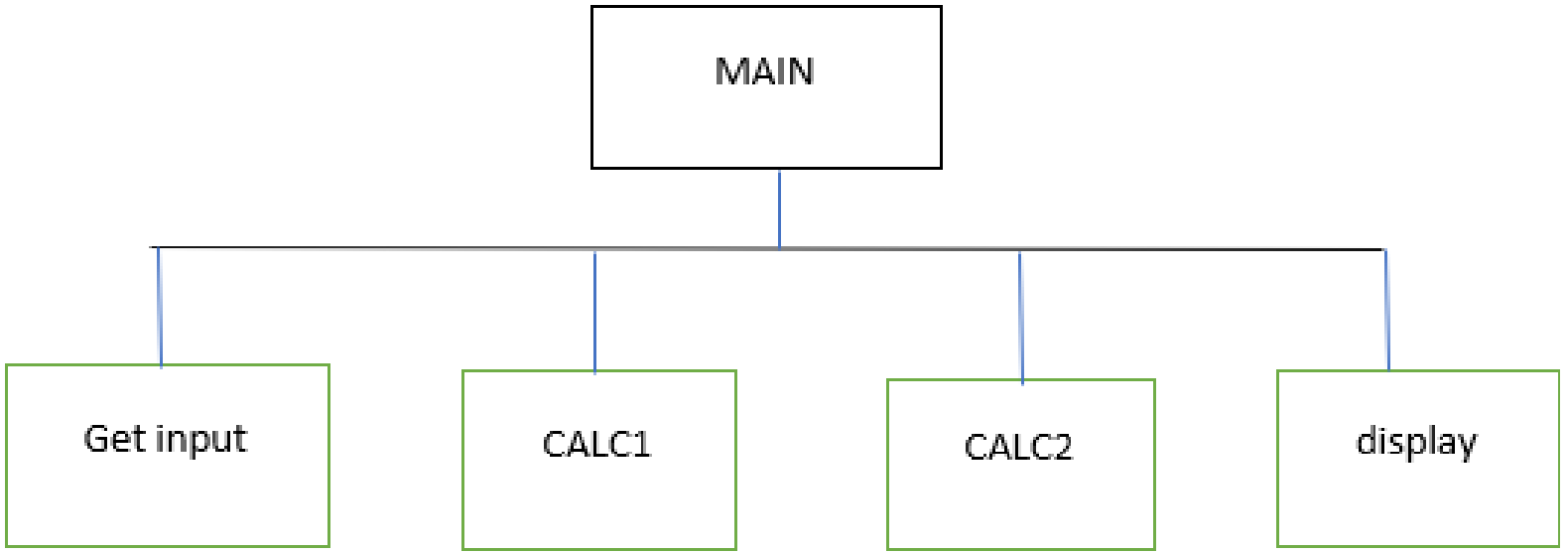
function stub format (example)

```
def calcamt(aa):
    pass
```

OR (better – for testing)

```
def calcamt(aa):
    print(‘ routine to be written’)
    return
```

consider



Example

```
def getinput():
    pass
def calc1():
    pass
def calc2():
    pass
def displayit():
    pass
def main():
    getinput()
    calc1()
    calc2()
    displayit()
    return
main()
```

Example

```
def getinput():
    print ('get input function')
def calc1():
    print('calculation 1 ')
def calc2():
    print ('calculation 2')
def displayit():
    print(' display the results')
def main():
    getinput()
    calc1()
    calc2()
    displayit()
    return
main()
```

Example

```
def getinput():
    print ('get input function')
    return
def calc1():
    print('calculation 1 ')
    return
def calc2():
    print ('calculation 2')
    return
del displayit():
    print(' display the results')
    return
```

Example

```
def main():
    ans= ‘y’
    while (ans == ‘y’ or ans ==‘Y’)
        getinput()
        calc1()
        calc2()
        displayit()
        ans = input(“Again (y/n): ”)
    return
main()
```

done