CLASS 08-10 – USER DEFINED FUNCTIONS

COMP130 – INTRODUCTION TO COMPUTING

DICKINSON COLLEGE

FUNCTIONS

- A ***function*** is a named sequence of statements that performs a computation.
  - We have used many functions/methods thus far:
    - `x=math.sin(2*math.pi)`
    - `y=random.randint(5,10)`
    - `time.sleep(2.5)`
    - `savings.deposit(100)`
    - `bob.forward(50)`
    - `c.move(5,0)`
- Sometimes we will find it useful to name a sequence of statements that we have written as well.
  - **Examples from UserDefFuncsExamples.ipynb**

USER DEFINED FUNCTIONS

- A ***function definition*** specifies the *name* of a new function and *the sequence of statements* that run when the function is *called* (or *invoked*).

```
def print_T():
    print('*****')
    print('  *  ')
    print('  *  ')
    print('  *  ')
    print('  *  ')
```

Function Header

Function Body

USING FUNCTIONS AND METHODS

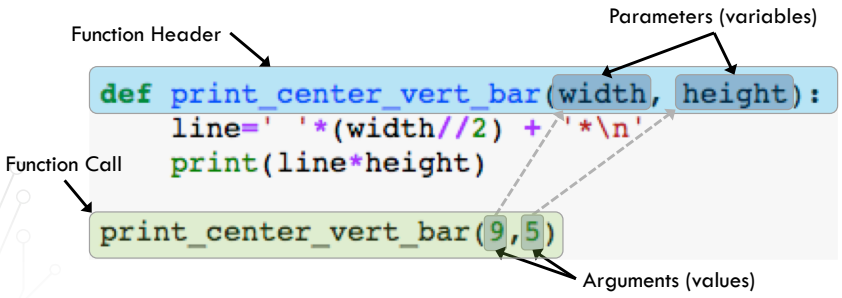
- Four Ways:
  - Built-in Functions
    - `print('This ' + str(1) + ' is built-in')`
  - Functions in a Module
    - `import math`
    - `dist = math.sqrt(5**2 + 9**2)`
  - Methods in an Object
    - `import bank`
    - `savings = bank.Account('Peixin', 1234)`
    - `savings.depost(100)`
  - User Defined Functions (following the definitions)
    - `print_T()`
    - `print_I()`

FUNCTIONS WITH ARGUMENTS

- ***Arguments*** are *evaluated* and the resulting values are *passed* to the *called/invoked* function to modify its behavior:
  - `print("Hello World!")`
  - `time.sleep(2.5)`
  - `savings.deposit(100)`
  - `bob.left(random.randint(45,135))`
  - `bob.forward(10*math.sin(math.degrees(math.pi)/2))`
- Arguments make functions:
  - More general (e.g. sine of any angle, sleep any time, deposit any amount).
  - Reusable
- User defined functions can also accept arguments .
  - **Examples from UserDefFuncsExamples.ipynb**

USER DEFINED FUNCTIONS: ARGUMENTS & PARAMETERS

- The values of the *arguments* given in the *function call* are assigned into the variables defined by the *parameters* listed in the *function header*.



THE FUNCTION CALL STACK

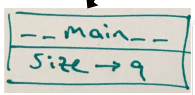
```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```

State Diagram



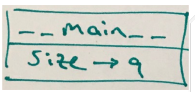
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



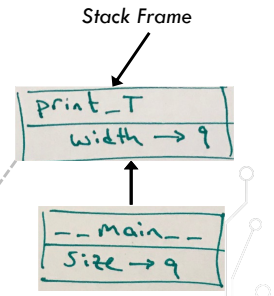
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



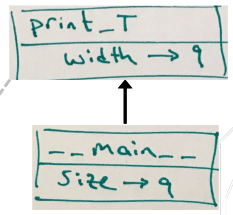
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



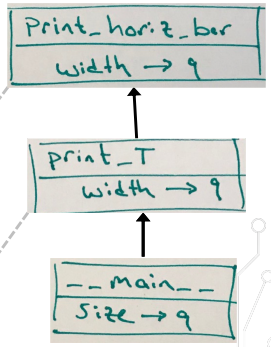
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



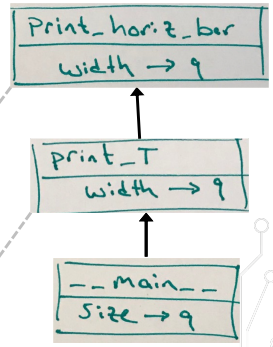
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



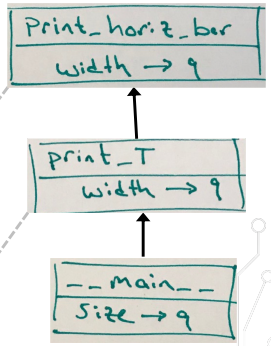
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



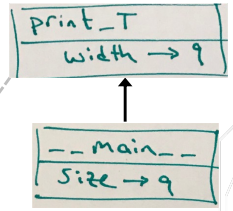
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



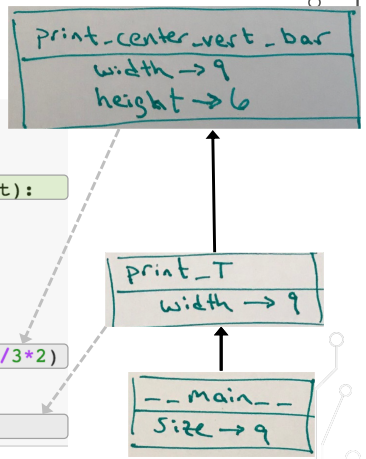
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



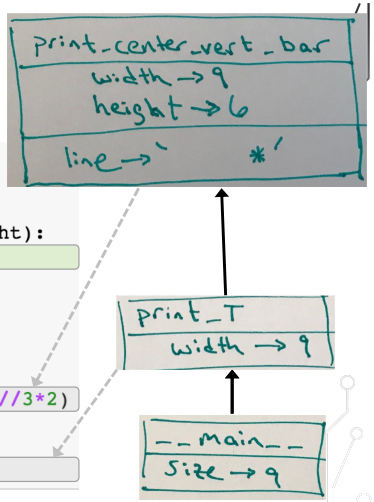
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



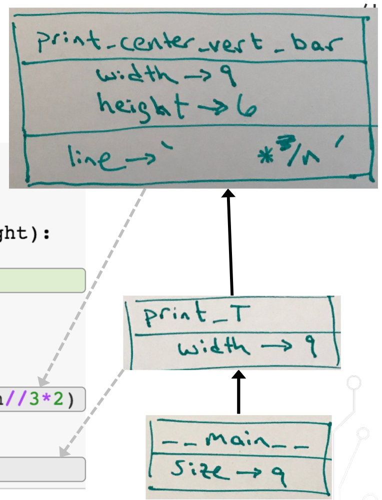
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



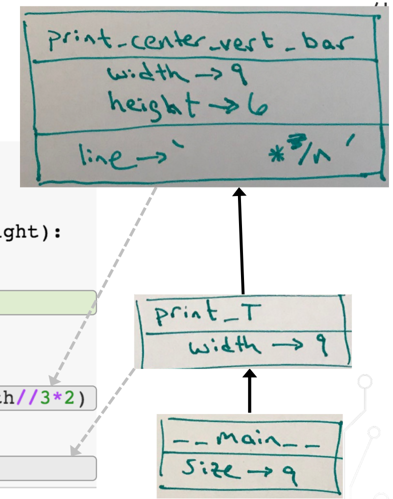
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



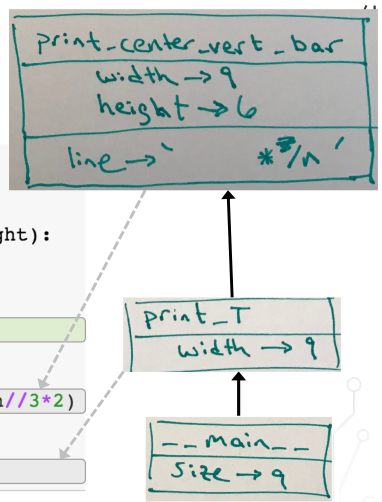
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



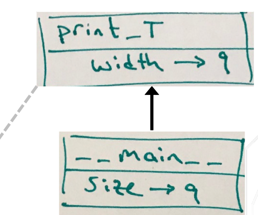
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



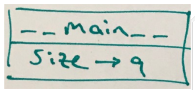
THE FUNCTION CALL STACK

```
def print_horiz_bar(width):
    print('*'*width)

def print_center_vert_bar(width, height):
    line=' '*(width//2)
    line=line + '\n'
    print(line*height)

def print_T(width):
    print_horiz_bar(width)
    print_center_vert_bar(width,width//3*2)

size=9
print_T(size)
```



THE FUNCTION CALL STACK

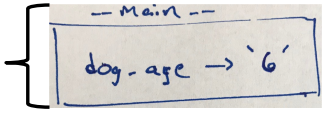
- The *function call stack* is a *transient* state diagram that shows the parameters and variables of the called/invoked functions.
  - A *stack frame* is added when a function is called and removed when it completes.
  - Parameters are *local*.
  - Variables created within a function are *local*.
- Python also uses the stack frame, behind the scenes, to track where to return to when the function completes.

FRUITFUL VS VOID (FRUITLESS?) FUNCTIONS

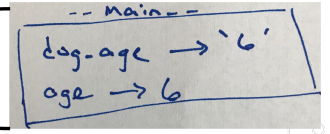
- Two distinct types of functions:
  - ***Fruitful functions*** return a value that is used in further computations.
    - `name_length = len("Ada Lovelace")`
    - `month_num = random.randint(1,12)`
    - `adj_len = hyp_len * math.cos(math.radians(angle_deg))`
    - `year = int(input("Enter the year"))`
  - ***Void (fruitless) functions*** perform some operation but *do not return a useful value.*
    - `print('John Von Neuman')`
    - `bob.forward(100)`
    - `sue.left(90)`
    - `print_horiz_bar(9)`
    - `print_I(9)`

A COMMON FRUITFUL FUNCTION MISTAKE

- A common mistake with some fruitful functions is to use them as if they modify their parameters.
  - `Example:`
    - `dog_age = input("Enter the dog's age:")`
      - Input returns a `string` so `dog_age` must be converted to a number to be used in computations.



      - Incorrect:
        - `float(dog_age)`
      - Correct:
        - `age=float(dog_age)`
        - OR: `dog_age = float(dog_age)`
        - OR: `dog_age = float(input("Enter the dog's age:"))`



THE NONE VALUE

- Void (fruitless) Functions perform some operation but ***do not return a useful value*.**
  - So we usually do not store that value or use it in an expression.
  - I.e. we don't write:
    - `name = print("Grace Murry Hopper")`
- They do return `None` (a "*useless*" *value*) though…
    - `print(name)`
      - `None`

WHY FUNCTIONS?

- **Readability:** Programs written using functions are easer for humans to read and understand and thus are also easier for us to maintain.
  - Ex: Reading `print_EAT` vs if it contains all of the individual `print` statements.
- **Maintainability:** Changing the behavior of a function in one place changes that behavior everywhere the function is called, making maintenance less error prone.
  - Ex. Printing all of our letters in X instead of *. Change functions, changes every program that uses them.
- **Problem Solving:** Functions provide a natural mechanism for breaking a big problem down into smaller easier to solve problems, and then putting the solution back together again (see Readability above).
  - Ex. `print_EAT` was a lot easer with `print_E`, `print_A` and `print_T` already done and tested.
- **Reusability:** Functions that are general can be written and tested once and then reused over and over in many programs (e.g. built-in functions and modules).
  - Ex. With all letters in hand, we could print any words we wanted in any program we want.