RECURSION

COMP130 – INTRODUCTION TO COMPUTING

DICKINSON COLLEGE

RECURSIVE FUNCTIONS

- A ***recursive function*** is a function that makes a call to itself.

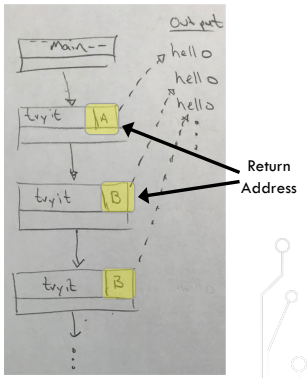
```
def tryit():
    print('hello')
    tryit()          # Line B

tryit()              # Line A
```

Recursive Function

Recursive Call

- A ***recursive call*** is a call to a function inside of it own body.
- ***Recursion*** is the process of executing recursive calls.
  - Managed with a stack diagram, no different than any other function calls.

Stack Diagram



BASE AND RECURSIVE CASES

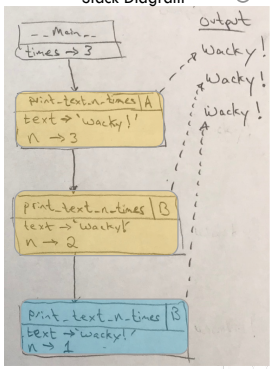
- Well formed recursive functions have:
  - One or more ***base cases*** where execution of the function does not make a recursive call.
  - One or more ***recursive cases*** where execution of the function makes a recursive call.

```
def print_text_n_times(text, n):
    if (n==1):
        print(text)
    else:
        print(text)
        print_text_n_times(text, n-1)    # Line B

times=3
print_text_n_times('wacky!', times)      # Line A
```

Base Case

Recursive Case

Stack Diagram

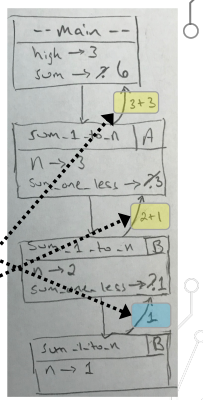


FRUITFUL RECURSIVE FUNCTIONS

- Recursive functions can return values.
  - E.g. `sum_1_to_n(10)` is `10 + sum_1_to_n(9)`

```
def sum_1_to_n(n):
    if n==1:
        return 1                         # Line C
    else:
        sum_one_less = sum_1_to_n(n-1)   # Line B
        return n + sum_one_less

high=3
sum = sum_1_to_n(high)                   # Line A
print(sum)
```

Stack Diagram



THINKING RECURSIVELY

i	fibonacci(i)
0	1
1	1
2	2
3	3
4	5
5	8
6	13
.	.
.	.
.	.

Base Case

fibonacci(i) = fibonacci(i-1)+ fibonacci(i-2)

Recursive Case

```
def fibonacci(i):
    if (i==0) or (i==1):
        return 1
    else:
        fib_m1 = fibonacci(i-1)
        fib_m2 = fibonacci(i-2)
        fib_i = fib_m1 + fib_m2
        return fib_i

fib_6=fibonacci(6)
print(fib_6)
```