Recursion

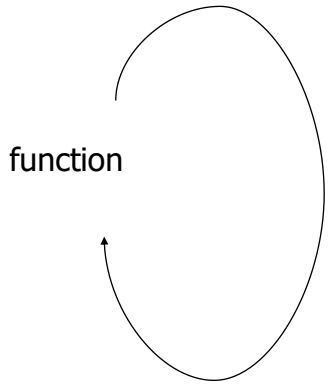
- A function calling itself directly or indirectly in a repetitive fashion.

Basic Definition Of Recursion

- *"A programming technique whereby a function calls itself either directly or indirectly."*

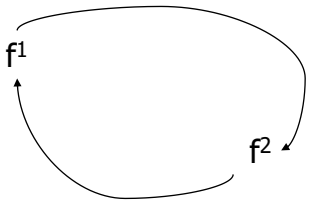
James Tam

Direct Call

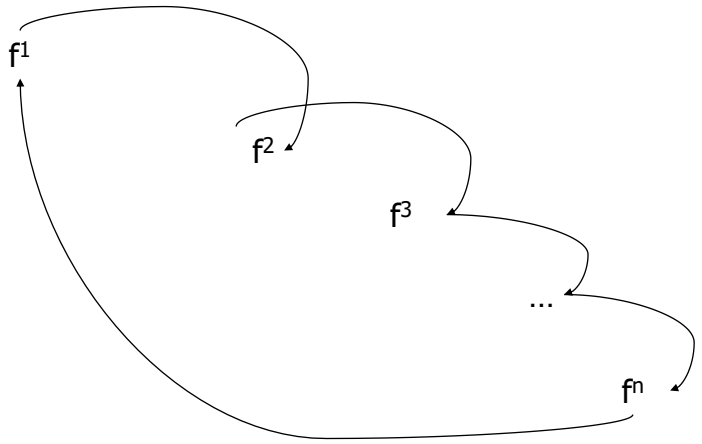



```
def fun ():
    ...
  fun ()
    ...
```

Indirect Call



Indirect Call



Indirect Call (2)

Name of the online example: `1simpleRecursive.py`

```
def fun1():
   print("\tfun1()")
   fun2()

def fun2():
   print("\tfun2()")
   fun1()

fun1()
```

Requirements For *Sensible* Recursion

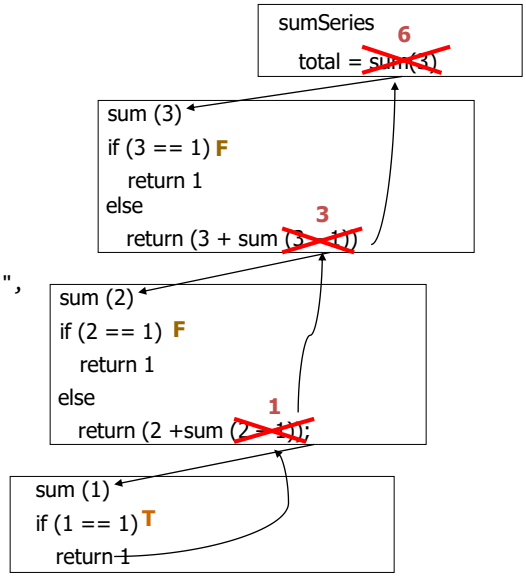
1) Base case
2) Progress is made (towards the base case)

Example Program: `2sumSeries.py`

```
def sum(no):
   if (no == 1):
      return 1
   else:
      return (no + sum(no-1) )


def start():
   last = input ("Enter the last
                  number: ")
   last = (int)last
   total = sum(last)
   print ("The sum of the series
           from 1 to", last, "is",
           total)

start()
```



When To Use Recursion

- When a problem can be divided into steps.
- The result of one step can be used in a previous step.
- There is a scenario when you can stop sub-dividing the problem into steps (step = recursive call) and return to a previous step.
  - Algorithm goes back to previous step with a partial solution to the problem (back tracking)
- All of the results together solve the problem.

When To Consider Alternatives To Recursion

- When a loop will solve the problem just as well
- Types of recursion (for both types a `return` statement is excepted)
  - **Tail recursion**
    - The last statement in the function is another recursive call to that function This form of recursion can easily be replaced with a loop.
  - **Non-tail recursion**
    - The last statement in the recursive function is not a recursive call.
      - Excludes t
    - This form of recursion is very difficult (read: impossible) to replace with a loop.

Example: Tail Recursion

- Tail recursion: A recursive call is the last statement in the recursive function.
- Name of the online example: `3tail.py`

```
def tail(no):
   if(no <= 3):
      print (no)
      tail(no+1)
   return()

tail(1)
```

Example: Non-Tail Recursion

- Non-Tail recursion: A statement which is not a recursive call to the function comprises the last statement in the recursive function.
- **Name of the online example**: `4nonTail.py`

```
def nonTail(no):
   if (no < 3):
      nonTail(no+1)
   print(no)
   return()

nonTail(1)
```

Error Handling Example Using Recursion

- **Name of the online example**: `5errorHandling_Loop.py`

  – Iterative/looping solution (month must be between 1 – 12)

```
JAN = 1
DEC = 12
month = -1
while((month < JAN) or (month > DEC)):
    month = int(input("Enter birth month (%d-%d): " \
                      %(JAN,DEC)))

print(month)
```

James Tam

Error Handling Example Using Recursion (2)

– **Name of the online example**: `6errorHandling_Recursive.py`

– Recursive solution (day must be between 1 – 31)

```
MIN = 1
MAX = 31
def promptDay():
    day = int(input("Enter day of birth (%d-%d): "
%(MIN,MAX)))
    if ((day < MIN) or (day > MAX)):
        day = promptDay()
    return(day)

day = promptDay()
print(day)
```

James Tam

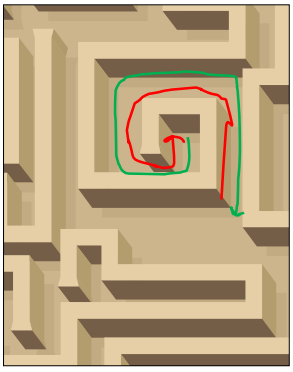
When To Use Iteration Or Recursion

- Rule of thumb for using iteration: if you can implement a solution using a loop then you should do so.
- When to employ a recursive solution: a loop cannot be employed.
  - "Back tracking" is needed.
  - Back tracking: When the repetition (whether via the iterations of a loop or a function calling itself over and over) ends the actual work of solving the problem occurs.
  - Examples: Traversing a maze, traversing a file system (folders/directories containing other folders).

James Tam

Applying Recursion: Traversing A Maze (Tutorial)

- Picked the wrong direction in the maze?
- After repeatedly traversing the maze (going up, left, right, down) and you hit a dead end!



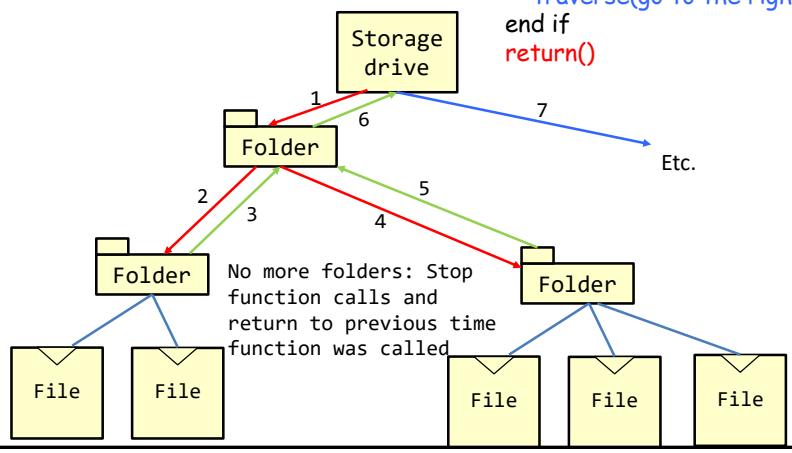
- You must "back track" (retrace your steps)

James Tam

Applying Recursion: Traversing A Directory/Folder Structure (Chart: James Tam)

Pseudo code

```
traverse(folder reference)
   If (reference leads a folder)
      traverse(go to left folder)
      traverse(go to the right folder)
   end if
   return()
```



Copyright Notification

- "Unless otherwise indicated, all images in this presentation are used with permission from Microsoft."

James Tam

You Should Now Know

- What is a recursive computer program
- How to write and trace simple recursive programs
- What are the requirements for recursion/What are the common pitfalls of recursion