**1.In mathematics, the Fibonacci numbers, commonly denoted Fn, form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1:**

The beginning of the sequence is this: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...  
The function fastFib(num) returns the fibonacci number Fn, of the given num as an argument.  
**Examples:**  
fib\_fast(5) ➞ 5  
fib\_fast(10) ➞ 55  
fib\_fast(20) ➞ 6765  
fib\_fast(50) ➞ 12586269025

In [1]:

*# Approach 1*

**def** fib\_fast\_one(in\_num):

temp\_list **=** []

**for** ele **in** range(in\_num**+**1):

**if** ele **in** [0,1]:

temp\_list**.**append(ele)

**else**:

temp\_list**.**append(temp\_list[**-**1]**+**temp\_list[**-**2])

print(f'fib\_fast\_one({in\_num}) ➞ {temp\_list[**-**1]}')

*# Approach 2 -> Memory Efficient*

**def** fib\_fast\_two(in\_num):

back\_two,back\_one,output **=** 0,1,0

**for** ele **in** range(in\_num**+**1):

**if** ele **>** 1:

output **=** back\_two**+**back\_one

back\_two **=** back\_one

back\_one **=** output

print(f'fib\_fast\_two({in\_num}) ➞ {output}')

fib\_fast\_one(5)

fib\_fast\_one(10)

fib\_fast\_one(20)

fib\_fast\_one(50)

print()

fib\_fast\_two(5)

fib\_fast\_two(10)

fib\_fast\_two(20)

fib\_fast\_two(50)

fib\_fast\_one(5) ➞ 5

fib\_fast\_one(10) ➞ 55

fib\_fast\_one(20) ➞ 6765

fib\_fast\_one(50) ➞ 12586269025

fib\_fast\_two(5) ➞ 5

fib\_fast\_two(10) ➞ 55

fib\_fast\_two(20) ➞ 6765

fib\_fast\_two(50) ➞ 12586269025

**2.Create a function that takes a strings characters as ASCII and returns each characters hexadecimal value as a string.**

**Examples:**  
convert\_to\_hex("hello world") ➞ "68 65 6c 6c 6f 20 77 6f 72 6c 64"  
convert\_to\_hex("Big Boi") ➞ "42 69 67 20 42 6f 69"  
convert\_to\_hex("Marty Poppinson") ➞ "4d 61 72 74 79 20 50 6f 70 70 69 6e 73 6f 6e"

In [2]:

**def** convert\_to\_hex(in\_string):

out\_string **=** []

**for** ele **in** in\_string:

out\_string**.**append(hex(ord(ele))[2:])

print(f'convert\_to\_hex({in\_string}) ➞ {" "**.**join(out\_string)}')

convert\_to\_hex("hello world")

convert\_to\_hex("Big Boi")

convert\_to\_hex("Marty Poppinson")

convert\_to\_hex(hello world) ➞ 68 65 6c 6c 6f 20 77 6f 72 6c 64

convert\_to\_hex(Big Boi) ➞ 42 69 67 20 42 6f 69

convert\_to\_hex(Marty Poppinson) ➞ 4d 61 72 74 79 20 50 6f 70 70 69 6e 73 6f 6e

**3.Someone has attempted to censor my strings by replacing every vowel with a \*, l\*k\* th\*s. Luckily, I've been able to find the vowels that were removed.**

Given a censored string and a string of the censored vowels, return the original uncensored string.  
**Examples:**  
uncensor("Wh\*r\* d\*d my v\*w\*ls g\*?", "eeioeo") ➞ "Where did my vowels go?"  
uncensor("abcd", "") ➞ "abcd"  
uncensor("\*PP\*RC\*S\*", "UEAE") ➞ "UPPERCASE"

In [3]:

**def** uncensor(in\_string,in\_vowels):

window **=** 0

out\_string **=** ''

**for** ele **in** in\_string:

**if** ele **==** '\*':

out\_string **+=** in\_vowels[window]

window **+=**1

**else**:

out\_string **+=** ele

print(f'uncensor{in\_string,in\_vowels} ➞ {out\_string}')

uncensor("Wh\*r\* d\*d my v\*w\*ls g\*?", "eeioeo")

uncensor("abcd", "")

uncensor("\*PP\*RC\*S\*", "UEAE")

uncensor('Wh\*r\* d\*d my v\*w\*ls g\*?', 'eeioeo') ➞ Where did my vowels go?

uncensor('abcd', '') ➞ abcd

uncensor('\*PP\*RC\*S\*', 'UEAE') ➞ UPPERCASE

**4.Write a function that takes an IP address and returns the domain name using PTR DNS records.**

**Examples:**  
get\_domain("8.8.8.8") ➞ "dns.google"  
get\_domain("8.8.4.4") ➞ "dns.google"

In [4]:

**import** socket

**def** get\_domain(in\_ip):

print(f'get\_domain({in\_ip}) ➞ {socket**.**gethostbyaddr(in\_ip)} ➞ {socket**.**gethostbyaddr(in\_ip)[0]}')

get\_domain("8.8.8.8")

get\_domain("8.8.4.4")

get\_domain(8.8.8.8) ➞ ('dns.google', [], ['8.8.8.8']) ➞ dns.google

get\_domain(8.8.4.4) ➞ ('dns.google', [], ['8.8.4.4']) ➞ dns.google

**5.Create a function that takes an integer n and returns the factorial of factorials. See below examples for a better understanding:**

**Examples:**  
fact\_of\_fact(4) ➞ 288  
# 4! \* 3! \* 2! \* 1! = 288  
fact\_of\_fact(5) ➞ 34560  
fact\_of\_fact(6) ➞ 24883200

In [5]:

**def** fact\_of\_fact(in\_num):

*# Internal Function to generate factorial of a Number*

**def** get\_factorial(n):

**if** n **==** 1:

**return** 1

**else**:

**return** n**\***get\_factorial(n**-**1)

out\_num **=** 1

**for** ele **in** range(1,in\_num**+**1):

out\_num **\*=** get\_factorial(ele)

print(f'fact\_of\_fact({in\_num}) ➞ {out\_num}')

fact\_of\_fact(4)

fact\_of\_fact(5)

fact\_of\_fact(6)

fact\_of\_fact(4) ➞ 288

fact\_of\_fact(5) ➞ 34560

fact\_of\_fact(6) ➞ 24883200