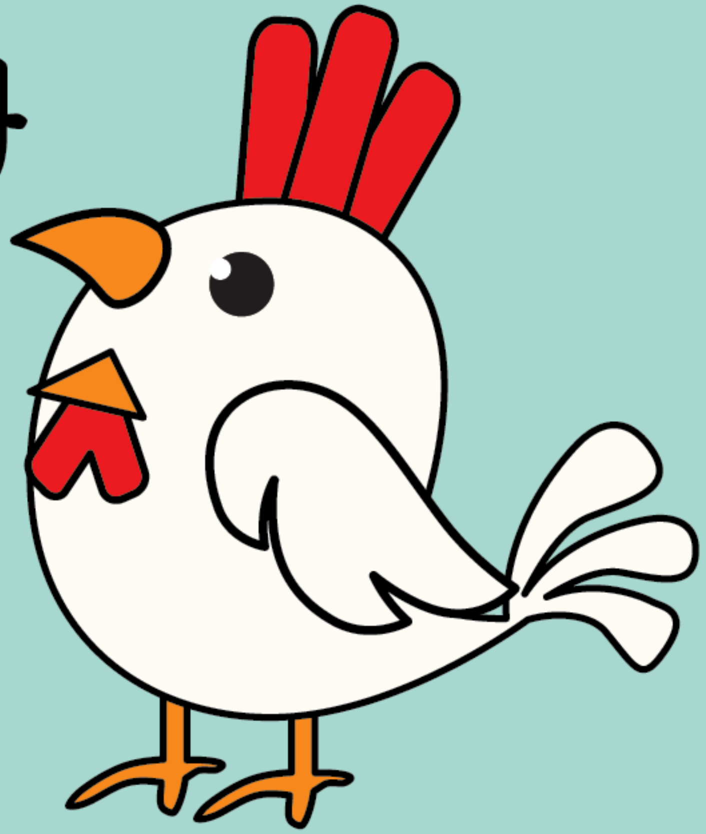
TMItalk: Too Much Information Talk

피보나치킨 수
피보나치 수열
그리고 황금비

치킨은
언제나
옳닭



- **피보나 치킨 수:**
  - $F(N)$명의 사람이 먹을 적당한 치킨 수는 $F(N-1)$마리이다.

N	1	2	3	4	5	6	7	8	9	10	11	12
$F(N)$	1	1	2	3	5	8	13	21	34	55	89	144

~~치킨~~ 제켄도르프의 정리: Zeckendorf's Theorem

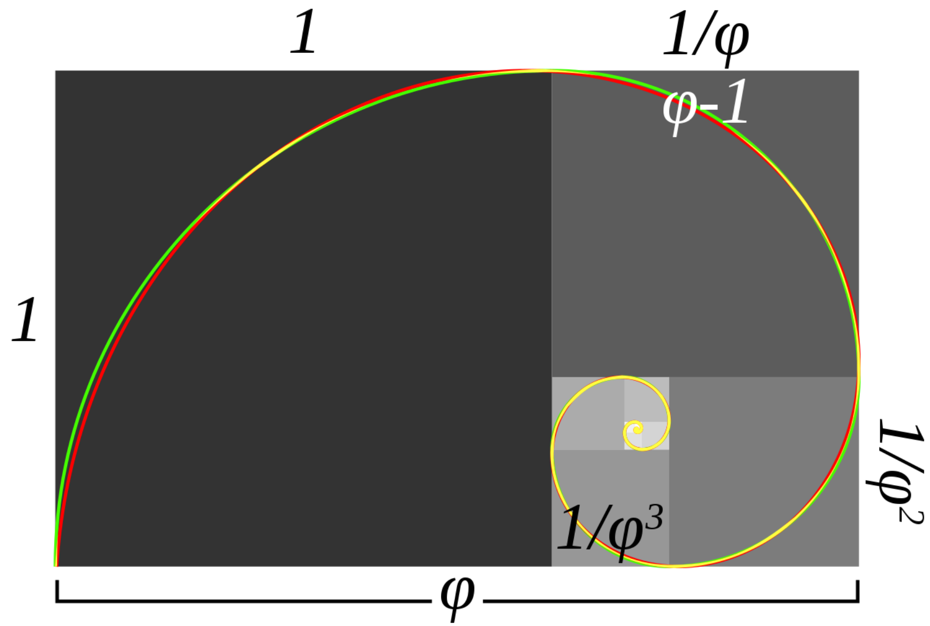
- 모든 자연수는 연속하지 않는 피보나치 수의 합으로 유일하게 표현할 수 있다.
  - $64 = 55 + 8 + 1 = F(10) + F(6) + F(1)$
  - 64의 피보나치킨 수 $= F(9) + F(5) + F(1) = 34 + 5 + 1 = 40$

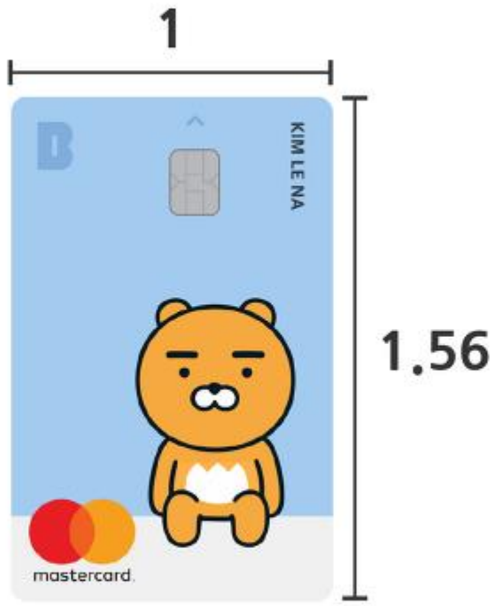
- 피보나 치킨 알고리즘 문제:
  - 임의의 자연수 N이 주어졌을 때,
    - 피보나치킨 수 $FibonaChicken(N)$을 구하는 알고리즘을 작성하라.
    - 위에서 작성한 알고리즘의 **시간 복잡도(time complexity)**를 분석하라.
    - 피보나치킨 문제의 **하한(lower bound)**을 분석하라.

- **FIBONA-CHICKEN-ALGORITHM**:
  - 입력: 임의의 자연수 N.
  - 출력: N명이 먹기에 적당한 치킨의 수 C.
  - 알고리즘:
    - 만약 N이 i번째 피보나치 수이면,
      - 즉, $N = F(i)$이면 $C = F(i-1)$을 리턴.
    - 만약 N이 피보나치 수가 아니면,
      - 제켄도르프 분해를 한다: $N = F(i_1) + F(i_2) + \cdots + F(i_k)$.
      - $C = F(i_1 - 1) + F(i_2 - 1) + \cdots + F(i_k - 1)$ 을 리턴.

- **피보나치 수에 대한 TMI:**
  - 피보나치 생성 함수: *Binet*의 공식
    - 임의의 자연수 i에 대하여, $N = F(i)$를 직접 계산할 수 있는가?
  - 피보나치 판별식:
    - 임의의 자연수 N에 대하여, N이 피보나치 수임을 판별할 수 있는가?
  - 피보나치 역함수:
    - 임의의 자연수 $N = F(i)$에 대하여, i를 직접 계산할 수 있는가?
  - 제켄도르프 분해:
    - 임의의 피보나치 수 $N = F(i)$에 대하여, 제켄도르프 분해를 할 수 있는가?

- **황금비**: Golden Ratio
  - **세상에서 가장 아름다운 비율이라고 알려져 있음**
    - $\varphi = \frac{1+\sqrt{5}}{2} = 1.618033988749895\cdots$
    - $\hat{\varphi} = \frac{1-\sqrt{5}}{2} = 1 - \varphi = -0.618033988749895\cdots$





▪ 피보나치 수와 황금비의 관계:

- 피보나치 수열에서 $F(i)$와 $F(i+1)$의 비율은 황금비에 수렴한다.

$$\lim_{n\to\infty} \frac{F(i+1)}{F(i)} = \varphi$$

$$\varphi = 1.618033988749895\cdots$$

$$\lim_{n\to\infty} \frac{F(i)}{F(i+1)} = -\hat{\varphi}$$

$$-\hat{\varphi} = 0.6180339887498945\cdots$$

```python
def FibonacciSequence(n):
    F = [0, 1]
    for i in range(2, n + 1):
        F.append(F[i - 1] + F[i - 2])
    return F

n = 50
F = FibonacciSequence(n)
for i in range(1, len(F) - 1):
    print(i, ":", F[i], F[i+1], F[i+1]/F[i])

from math import sqrt
phi = (1 + sqrt(5)) / 2
phi_ = 1 - phi
print(phi)
print(phi_)
```

```
1 : 1 1 1.0
2 : 1 2 2.0
3 : 2 3 1.5
4 : 3 5 1.6666666666666667
5 : 5 8 1.6
6 : 8 13 1.625
7 : 13 21 1.6153846153846154
8 : 21 34 1.619047619047619
9 : 34 55 1.6176470588235294
10 : 55 89 1.6181818181818182
……
31 : 1346269 2178309 1.6180339887496482
32 : 2178309 3524578 1.618033988749989
33 : 3524578 5702887 1.618033988749859
34 : 5702887 9227465 1.6180339887499087
35 : 9227465 14930352 1.6180339887498896
36 : 14930352 24157817 1.618033988749897
37 : 24157817 39088169 1.618033988749894
38 : 39088169 63245986 1.6180339887498951
39 : 63245986 102334155 1.6180339887498947
40 : 102334155 165580141 1.618033988749895
……

1.618033988749895
-0.6180339887498949
```

▪ 비네의 공식: Binet's Formula

- 1843년, 프랑스의 수학자 Binet가 발견한 피보나치 수의 생성 함수
  - 임의의 자연수 i에 대하여, $F(i) = \frac{\varphi^i - \hat{\varphi}^i}{\varphi - \hat{\varphi}} = \frac{1}{\sqrt{5}}(\varphi^i - \hat{\varphi}^i)$.

```
from math import sqrt, floor

def Binet(i):
    return round((phi ** i - phi_ ** i) / sqrt(5))

n = 50
for i in range(n + 1):
    print(i, Binet(i))
```

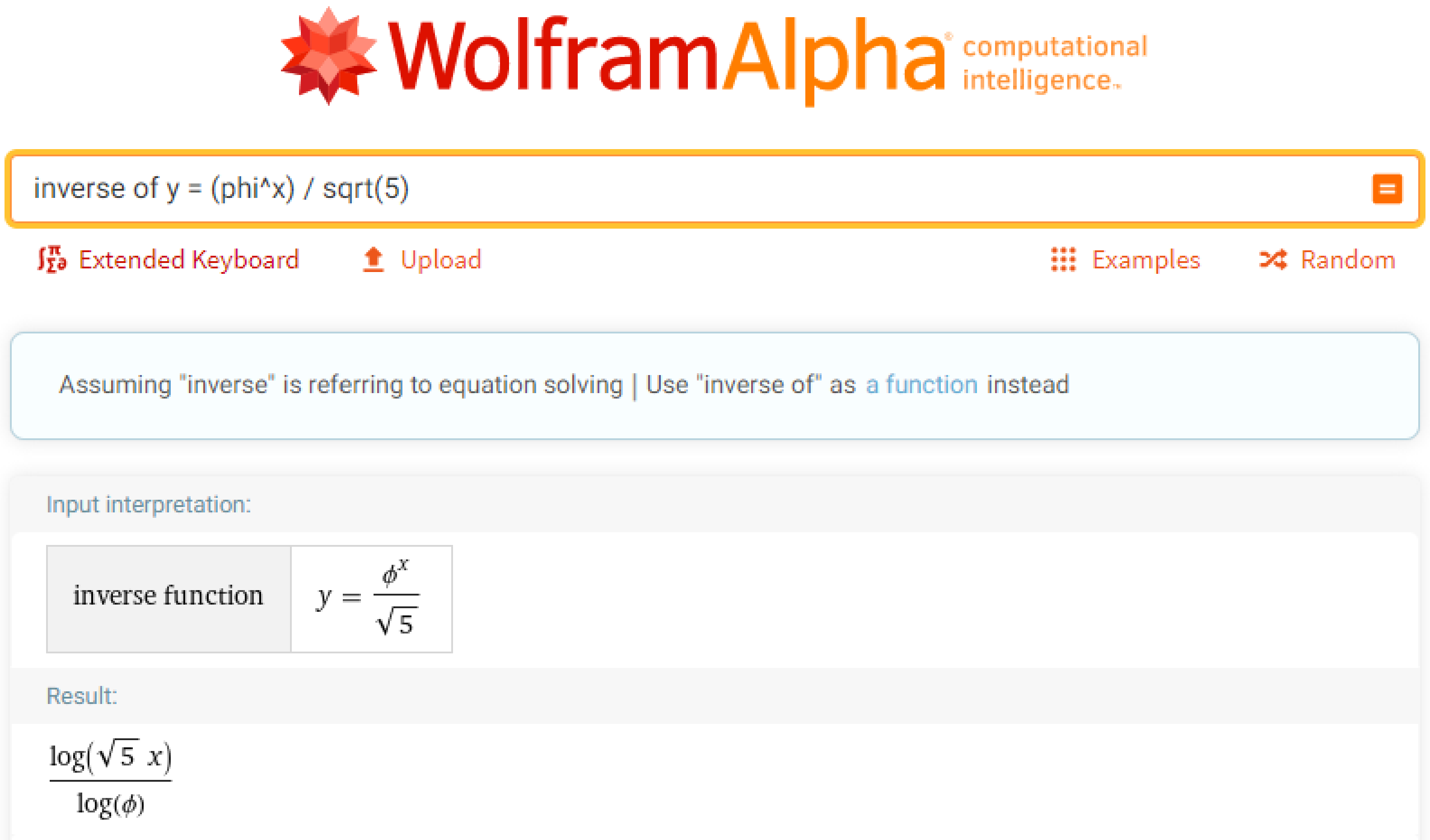
- **피보나치 수의 판별식:**
  - Binet의 공식을 응용하면 다음을 증명할 수 있다.
    - 자연수 N이 피보나치 수일 필요충분조건은 다음과 같다.
    - $5N^2 + 4$, 또는, $5N^2 - 4$가 **완전제곱수**(perfect square)이다.
  - 상세한 증명은 Math Storehouse 블로그 참고:
    - https://mathstorehouse.com/archives/mathematics/algebra/number-theory/2232

```python
from math import sqrt, floor

def is_perfect(n):
    rootn = floor(sqrt(n))
    if rootn * rootn == n:
        return True
    return False

def is_fibonacci(N):
    x, y = 5 * N * N + 4, 5 * N * N - 4
    return is_perfect(x) or is_perfect(y)

N = 10000000
F = []
for i in range(N + 1):
    if is_fibonacci(i):
        F.append(i)
print(F)
```

```
[0, 1, 2, 3, 5, 8, 13, 21, 34, 55,
89, 144, 233, 377, 610, 987, 1597,
2584, 4181, 6765, 10946, 17711,
28657, 46368, 75025, 121393,
196418, 317811, 514229, 832040,
1346269, 2178309, 3524578,
5702887, 9227465]
```

- **피보나치 수의 역함수:**
  - 피보나치 수 $F(i)$가 주어졌을 때, i의 값을 알려면,
    - Binet의 공식에서 역함수를 취하면 된다.
  - 역함수는 어떻게 구하지? 울프럼 알파가 있잖아!:
    - https://www.wolframalpha.com/
    - 역함수 구하기: inverse of y = (phi^x) / sqrt(5)

TMITalk: 피보나치킨 수, 피보나치 수열, 그리고 황금비



```
from math import sqrt, log

phi = (1 + sqrt(5)) / 2

def inverse_fibonacci(N):
    return round(log(sqrt(5) * N) / log(phi))

def FibonacciSequence(n):
    F = [0, 1]
    for i in range(2, n + 1):
        F.append(F[i - 1] + F[i - 2])
    return F

n = 50
F = FibonacciSequence(n)
for i in range(1, len(F)):
    print(i, inverse_fibonacci(F[i]))
```

```
1 2
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
11 11
12 12
13 13
14 14
15 15
16 16
17 17
......
```

- **피보나치킨 수:**
  - N이 피보나치 수이면 O(1)이면 쌉가능!

```
def FibonaChicken(N):
    if is_fibonacci(N):
        return Binet(inverse_fibonacci(N) - 1)

N = int(input("자애로운 자여, 몇 명이나 먹이려고 하는고? "))
C = FibonaChicken(N)
print("그렇다면", C, "마리를 시키거라")
print("능히", N, "명을 먹이는데 부족함이 없느니라.")
```

피보나치킨 수:

- N이 피보나치 수가 아니면: 제켄도르프 분해로 $O(\frac{k}{2}) = O(k)$에 쌉가능!

```
def FibonaChicken(N):
    if N <= 2:
        return 1
    i = inverse_fibonacci(N)
    while N > Binet(i):
        i += 1
    return Binet(i - 2) + FibonaChicken(N - Binet(i - 1))
```

- **피보나치 수**: The Final Solution

```python
def FibonaChicken(N):
    if N <= 2:
        return 1
    i = inverse_fibonacci(N)
    if is_fibonacci(N):
        return Binet(i - 1)
    else:
        while N > Binet(i):
            i += 1
        return Binet(i - 2) + FibonaChicken(N - Binet(i - 1))
```

- **피보나치킨 알고리즘의 복잡도 분석:**
  - FIBONA-CHICKEN-ALGORITHM 은
    - N이 피보나치 수일 때 $O(1)$에 가능
    - N이 피보나치 수가 아니면 $O(k)$에 가능
      - 여기서 k는 $F(N)$보다 작거나 같은 피보나치 수열의 길이


Any Questions?
주니온TV@Youtube
자세히 보면 유익한 코딩 채널