

# Rekurencja

Żeby zrozumieć rekurencję,  
najpierw musisz zrozumieć rekurencję.

# Rekurencja

Rekurencja, zwana także rekursją – odwoływanie się np. funkcji lub definicji do samej siebie.<sup>1</sup>

Każda definicja rekurencyjna potrzebuje przynajmniej jednego przypadku bazowego (nie rekurencyjnego). W przeciwnym wypadku nigdy się nie zakończy (**warunek stopu**).<sup>1</sup>

*Innymi słowy:*

W pewnym momencie wywoływanie funkcji przez samą siebie musi się kiedyś zakończyć, dlatego potrzebny jest stan, w którym dana funkcja nie jest wywoływana ponownie.

<sup>1</sup>: <https://pl.wikipedia.org/wiki/Rekurencja>



# Przykład - Rekurencja w Google

Google

rekurencja

Wszytko Grafika Filmy Wiadomości Zakupy Więcej Ustawienia

Okolo 56 700 wynikow (0,36 s)

Czy chodzilo Ci o: **rekurencja**

**Rekurencja**, zwana takze rekursja (ang. recursion, z lac. recurrere, przybiec z powrotem) – odwoływanie się np. funkcji lub definicji do samej siebie.

**Rekurencja – Wikipedia, wolna encyklopedia**  
<https://pl.wikipedia.org/wiki/Rekurencja>



Rekurencja

# Przykład - Suma kolejnych liczb całkowitych

**W matematyce:**

$$f(0) = 0$$

$$f(n) = n + f(n - 1)$$



Rekurencja

# Przykład - Suma kolejnych liczb całkowitych

$$f(0) = 0$$

$$f(n) = n + f(n - 1)$$

Rozwinięcie:


$$\begin{aligned} f(5) &= 5 + f(5 - 1) \\ &= 5 + f(4) = 5 + (4 + f(4 - 1)) \\ &= 5 + (4 + f(3)) = 5 + (4 + (3 + f(3 - 1))) \\ &= 5 + (4 + (3 + f(2))) = 5 + (4 + (3 + (2 + f(2 - 1)))) \\ &= 5 + (4 + (3 + (2 + f(1)))) = 5 + (4 + (3 + (2 + (1 + f(1 - 1))))) \\ &= 5 + (4 + (3 + (2 + (1 + f(0))))) = 5 + (4 + (3 + (2 + (1 + 0)))) \\ &= 15 \end{aligned}$$



Rekurencja

# Przykład - Suma kolejnych liczb całkowitych

To samo zapisane **w języku Python.**

```
def f(n):  
    if n == 0:  warunek stopu  
        return 0  
    else:  
        return n + f(n - 1)  
  
x = f(5)  
  
print(x)
```



# Przykład - Suma kolejnych liczb całkowitych

Rozwinięcie dla języka Python:

```
x = f(5)
    = 5 + f(5 - 1)
    = 5 + f(4) = 5 + (4 + f(4 - 1))
    = 5 + (4 + f(3)) = 5 + (4 + (3 + f(3 - 1)))
    = 5 + (4 + (3 + f(2))) = 5 + (4 + (3 + (2 + f(2 - 1))))
    = 5 + (4 + (3 + (2 + f(1)))) = 5 + (4 + (3 + (2 + (1 + f(1 - 1)))))
    = 5 + (4 + (3 + (2 + (1 + f(0))))) = 5 + (4 + (3 + (2 + (1 + 0))))
    = 5 + (4 + (3 + (2 + (1))))
    = 5 + (4 + (3 + (3)))
    = ...
```



Rekurencja

# Rekurencja

## Przykład - Suma kolejnych liczb całkowitych (Python)

= ...

= 5 + (4 + **(6)**)

= 5 + (**10**)

= 15





# Przykład - Ciąg Fibonacciego

**W matematyce:**

$$F(0) = 0$$

$$F(1) = 1$$

$$F(n) = F(n-1) + F(n-2)$$



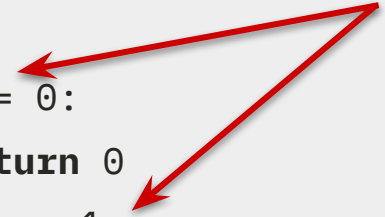
Rekurencja

# Przykład - Ciąg Fibonacciego

To samo zapisane **w języku Python.**

```
def F(n):  
    if n == 0:  
        return 0  
    elif n == 1:  
        return 1  
    else:  
        return F(n - 1) + F(n - 2)
```

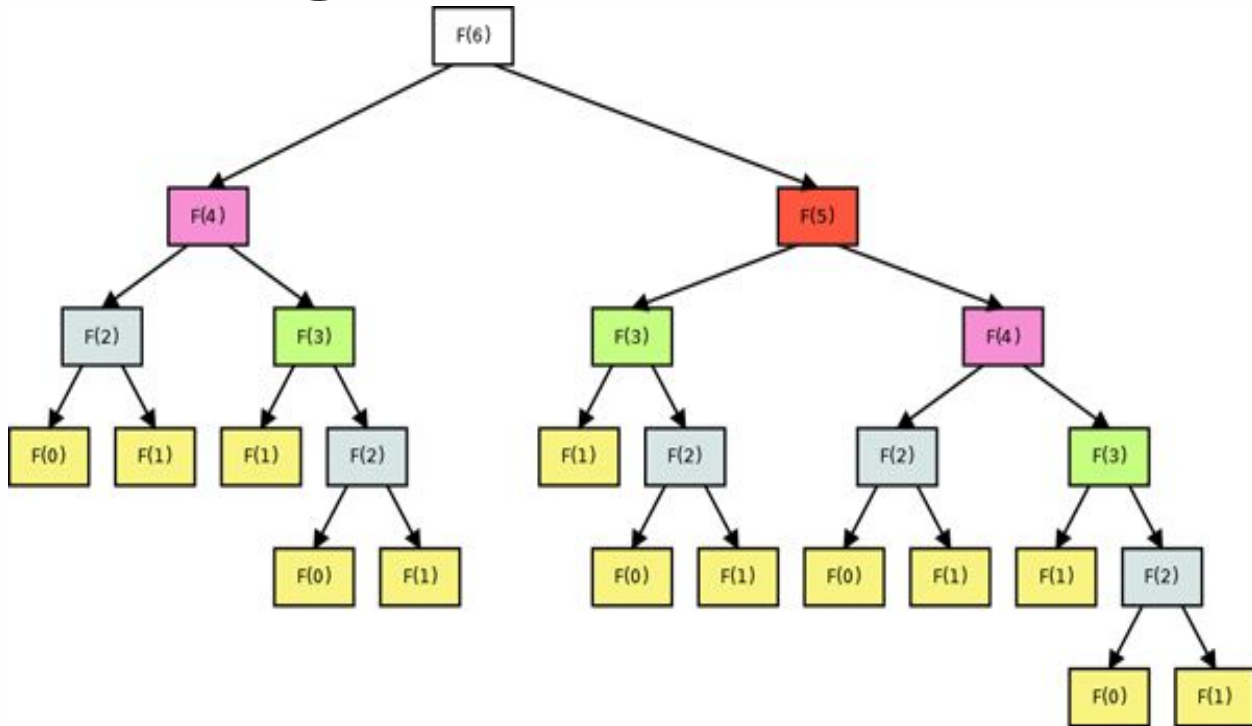
warunek stopu



## Przykład - Ciąg Fibonacciego

Obliczenia wykonywane rekurencyjnie **nie zawsze są efektywne.**

Obliczenia rekurencyjne często wymagają większej liczby wykonywanych obliczeń, a także wiążą się ze zwiększonym wykorzystaniem pamięci.



Źródło diagramu: <http://home.agh.edu.pl/~pkleczek/dokuwiki/doku.php?id=dydaktyka:aisd:2016:recursion>  
 Autor: Paweł Kleczek

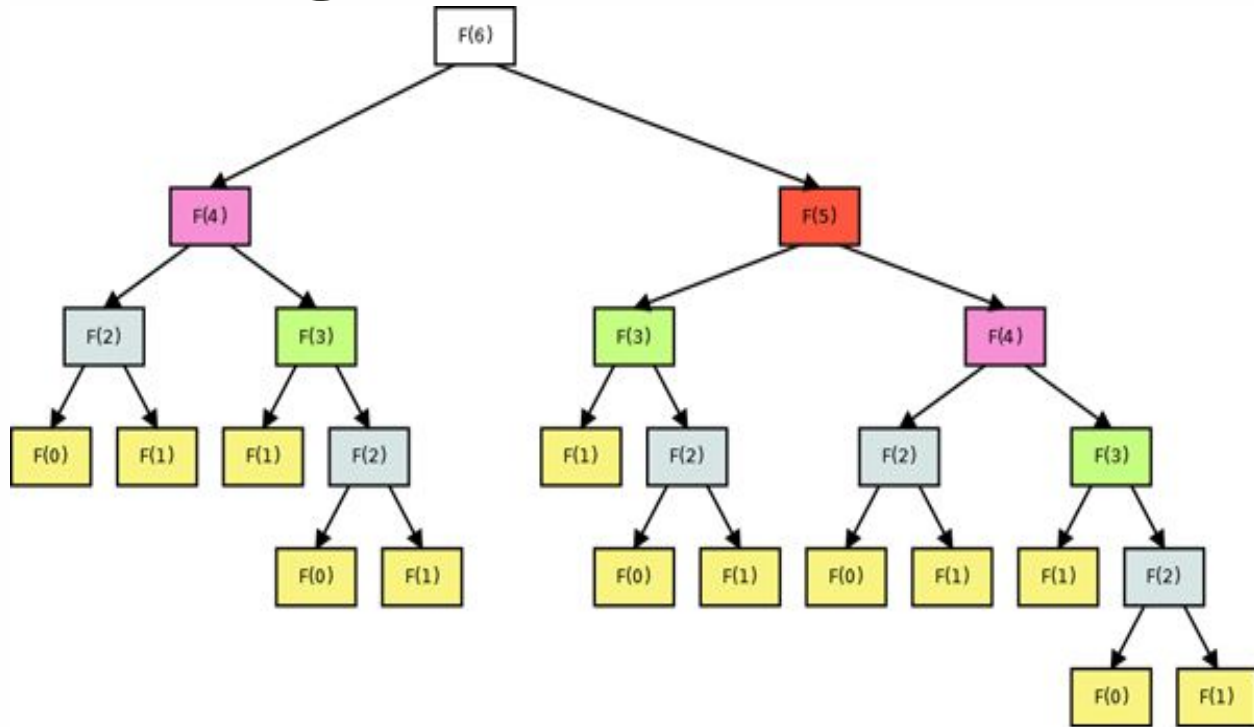


## Rekurencja

# Przykład - Ciąg Fibonacciego

Obliczenia wykonywane rekurencyjnie **nie zawsze są efektywne**.

Po prawej stronie znajduje się drzewo wywołań funkcji  $F$ . Okazuje się, że ta funkcja jest wywoływana wiele razy w celu obliczenia tej samej wartości (np.  $F(2)$ ).



Źródło diagramu: <http://home.agh.edu.pl/~pklecze/dokuwiki/doku.php?id=dydaktyka:aisd:2016:recursion>  
Autor: Paweł Kleczek



# Pytania

1. Czym jest rekurencja?
2. Czym jest warunek stopu?
3. Dlaczego zastosowanie rekurencji nie zawsze jest efektywne?



Rekurencja

# Literatura

1. Rekurencja, Education in Mathematics and Computing, Uniwersytet w Waterloo,  
<https://cscircles.cemc.uwaterloo.ca/16-pl/>
2. Rekurencja, Paweł Kłeczek,  
<http://home.agh.edu.pl/~pkleczek/dokuwiki/doku.php?id=dydaktyka:aisd:2016:recursion>



