CYFROWA SZKOŁA
Wielkopolsk@ 2020

Funkcje rekurencyjne

Wprowadzenie do języka Python (XV)

CYFROWA SZKOŁA
Wielkopolsk@ 2020

FUNKCJE REKURENCYJNE

Funkcja rekurencyjna to funkcja, która wywołuje samą siebie. Możesz zadać sobie pytanie, dlaczego miałbyś kiedykolwiek chcesz to zrobić? W rzeczywistości niektóre problemy można łatwiej rozwiązać poprzez rekurencję.

Każda funkcja rekurencyjna musi zawierać warunek, który zatrzymuje proces wywoływanie funkcji.

Rozwiązujemy problem, wykonując kilka prostych powtarzalnych obliczeń w porównaniu z pisaniem jednego dużego bloku kodu.

Jako przykład – obliczanie silni.

CYFROWA SZKOŁA
Wielkopolsk@ 2020

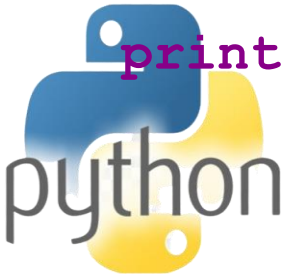
Obliczanie silni

Obliczanie silni jest zwykle wykonywane za pomocą metody rekurencyjnej

funkcja 3! = 3 * 2 * 1

KOD

```
def silnia(liczba):
    # Każda funkcja rekurencyjna musi zawierać warunek
    # kiedy przestanie się wywoływać (zakończy się)
    if liczba <= 1:
        return 1
    else:
        wynik = liczba * silnia(liczba - 1)
        return wynik
print(silnia(4))
```



CYFROWA SZKOŁA
Wielkopolsk@ 2020

Jak to działa

1-szy : wynik = 4 * silnia(3) = 4 * 6 = 24

2. : wynik = 3 * silnia(2) = 3 * 2 = 6

3: wynik = 2 * silnia(1) = 2 * 1 = 2

Aby rozwiązać ten problem, pracujemy w dół wywołań drabiny obliczeń aż do funkcji silnia, kiedy otrzymuje wartość 1. Kiedy to nastąpi podczas trzeciego przejścia, możemy zakończyć nasz pierwsze obliczenia uzyskując wynik 2. Następnie przesuwamy 2 w górę drabiny, aby rozwiązać następne obliczenie (3 * 2 = 6). Następnie 6 przesuwa się w górę, aby zakończyć obliczenia (4 * 6 = 24)

CYFROWA SZKOŁA
Wielkopolsk@ 2020

Problem z Pythonem do rozwiązania

Aby przetestować to, czego się nauczyłeś, chcę teraz, abyś obliczył liczby Fibonacciego.

Aby obliczyć liczby Fibonacciego, sumujemy 2 poprzednie wartości, aby obliczyć następny element m tworząc ciąg w ten sposób 1, 1, 2, 3, 5, 8 ...

Ciąg Fibonacciego jest zdefiniowany przez:

$F_n = F_{n-1} + F_{n-2}$

Gdzie $F_0 = 0$ i $F_1 = 1$

Oto przykład, który może pomóc

print(fib(3))

1-szy : wynik = fib(2) + fib(1) : 2 + 1

2. : wynik = (fib(1) + fib(0)) + (fib(0)) : 1 + 0

3: wynik = fib(2) + fib(1)

print(fib(4))

1-szy : wynik = fib(3) + fib(2) : 3 + 2

2. : wynik = (fib(2) + fib(1)) + (fib(1) + fib(0)) : 2 + 1

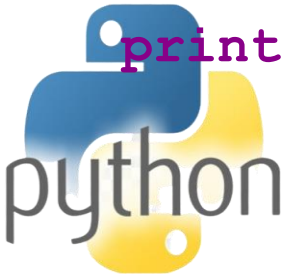
3: wynik = (fib(1) + fib(0)) + fib(0) : 1 + 0

Spróbuj. Celem jest nakłonienie Cię do myślenia w nowy sposób i zrozumienia końcowego wyniku.

CYFROWA SZKOŁA
Wielkopolsk@ 2020

Rozwiązanie

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        wynik = fib(n-1) + fib(n-2)
        return wynik
print(fib(3))
print(fib(4))
```



CYFROWA SZKOŁA
Wielkopolsk@ 2020

Drugi problem Pythona do rozwiązania

Wcześniej wygenerowaliśmy 1 liczbę w ciągu Fibonacciego. Tym razem poproś użytkownika o zdefiniowanie ile liczb chcą i wyświetlają ich.

Pamiętaj, że wzór na obliczenie ciągu Fibonacciego to

$F_n = F_{n-1} + F_{n-2}$

Gdzie $F_0 = 0$ i $F_1 = 1$

CYFROWA SZKOŁA
Wielkopolsk@ 2020

Oto przykładowe dane wyjściowe, do których powinieneś dążyć

Ile wartości Fibonacciego należy znaleźć : 5

1

1

2

3

5

Wszystko gotowe

Użyjesz tej samej funkcji powyżej z pętlą while dla wyjścia.

CYFROWA SZKOŁA
Wielkopolsk@ 2020

Rozwiązanie

```
def fib(liczba):
    if liczba == 0:
        return 0
    elif liczba == 1:
        return 1
    else:
        wynik = fib(liczba - 1) + fib(liczba - 2)
        return wynik
```


```
num_fib_vals = int(input("Ile wartości Fibonacciego należy znaleźć:
"))
i = 1
# Chociaż i jest mniejsze niż liczba żądanych wartości
# kontynuuj, aby znaleźć więcej
while i < num_fib_vals:
    # Wywołaj fib()
    fib_val = fib(i)
    print(fib_val)
    i += 1
print("Wszystko gotowe")
```