

Programowanie proceduralne

INP001210WL

rok akademicki 2021/22

semestr letni

Wykład 1

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220



Plan prezentacji (1)

- Co to jest algorytm?
- Co to jest program?

- Funkcje - powtórzenie
 - definiowanie i wywoływanie
 - funkcja `main()`
 - zmienne lokalne
 - przekazywanie argumentów
 - zwracanie wartości



Plan prezentacji (2)

- Funkcje rekurencyjne
 - Co to jest rekurencja?
 - Warunek stopu
 - Przykłady algorytmów rekurencyjnych



Co to jest algorytm?

- Algorytm jest przepisem opisującym krok po kroku rozwiązanie problemu lub osiągnięcie jakiegoś celu
- Algorytmika to dziedzina zajmująca się algorytmami i ich właściwościami

Algorytm

- Słowo algorytm wywodzi się od nazwiska arabskiego matematyka i astronoma. Muhammad ibn Musa al-Chuwarizmi żył na przełomie VIII i IX wieku, jest uznawany za prekursora metod obliczeniowych w matematyce



https://pl.wikipedia.org/wiki/Muhammad_ibn_Musa_al-Chuwarizmi



Przykład algorytmu

KAKAOWE CIASTO BEZ PIECZENIA

POPULARNE CIASTA

SZYBKIE CIASTO

DESERY BEZ PIECZENIA

CIASTA

CIASTA CZEKOLADOWE

CIASTA KOSTKI

CZEKOLADA

Rewelacyjne ciasto bez pieczenia! Kakaowe hebatniki przełożone kakaową masą budyniową i dżemem porzeczkowym (można połączyć pół na pół z powidłami śliwkowymi).



Przykład algorytmu

SKŁADNIKI

KREM CZEKOLADOWY

1 litr mleka

3 łyżki mąki pszennej

3 łyżki mąki ziemniaczanej

1 szklanka cukru

3 łyżki kakao

2 żółtka

ORAZ

ok. 600 g kakaowych herbatników
"petit beurre"

200 g masła

1 słoiczek dżemu porzeczkowego
lub powideł śliwkowych

kakao

PRZYGOTOWANIE

DODAJ NOTATKĘ



KREM CZEKOLADOWY

- Odląć 1 i 1/2 szklanki mleka i dokładnie wymieszać je (np. różgą) z mąką pszenną i ziemniaczaną, cukrem, żółtkami oraz likierami jeśli ich używamy.
- Resztę mleka zagotować (dokładnie, aż zaczną kipieć), następnie wlewać do niego mieszankę mleka, mąki i żółtek, jednocześnie energicznie mieszając różgą. Zagotować co chwilę mieszając.
- Po zagotowaniu gotowy budyń odstawić z ognia, przelać do czystej miski i całkowicie ostudzić (na wierzch można położyć folię spożywczą aby nie zrobił się kożuch).
- Miękkie masło ubijać przez ok. 3 minuty aż się napuszy, następnie stopniowo, w krótkich odstępach czasu, dodawać budyń ciągle ubijając.

Przykład algorytmu

PRZEŁOŻENIE

- Formę o wymiarach ok. **20 x 30 cm** (może być większa) wysmarować masłem i wyłożyć papierem do pieczenia. Układać warstwami na przemian herbatniki i krem budyniowy, otrzymując 4 lub 5 takich warstw, druga warstwa od dołu ma mieć zamiast kremu - dżem. Na wierzchu ma być cienka warstwa kremu.
- Ciasto oprószyć kakao i wstawić do lodówki na kilka godzin lub noc. Ciasto można przygotować dzień wcześniej.

WSKAZÓWKI

Opcjonalnie do kremu można dodać 4 łyżki likieru pomarańczowego lub amaretto lub orzechowego lub 2 łyżki mocnego alkoholu.





Algorytm Euklidesa

- Algorytm wyznaczania największego wspólnego dzielnika dwóch dodatnich liczb całkowitych został sformułowany w IV w. p.n.e. przez Euklidesa



Co to jest program?

- Program to lista szczegółowych instrukcji przekazywanych komputerowi do realizacji określonych zadań



Program i przepis a algorytm





Funkcje - powtórzenie

Korzyści z dzielenia programu na funkcje:

- czytelniejszy kod
- wielokrotne wykorzystanie kodu
- lepsze testowanie (łatwiej testować podzadanie umieszczone w osobnej funkcji)
- szybsze tworzenie oprogramowania
- łatwiejsza praca w zespołach

Funkcje - powtórzenie

Definiowanie i wywoływanie

- Ogólny schemat definicji funkcji

```
def nazwa_funkcji():
```

```
    polecenie
```

```
    polecenie
```

```
    itd.
```



nagłówek funkcji

blok poleceń

Funkcje - powtórzenie

Definiowanie i wywoływanie

- Przykładowa definicja funkcji


```
def message():
```

```
    print('Jestem Artur,')
```

```
    print('król Brytyjczyków.')
```



nagłówek funkcji



blok
poleceń

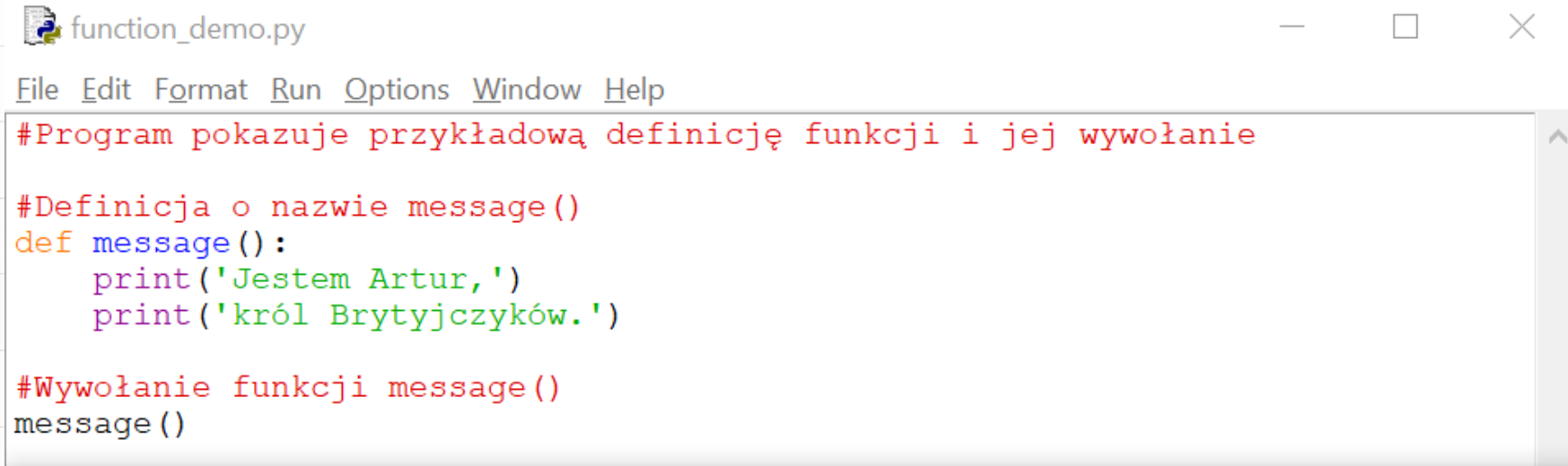
Funkcje - powtórzenie

Definiowanie i wywoływanie

- Wywołanie funkcji
`message ()`

Funkcje - powtórzenie

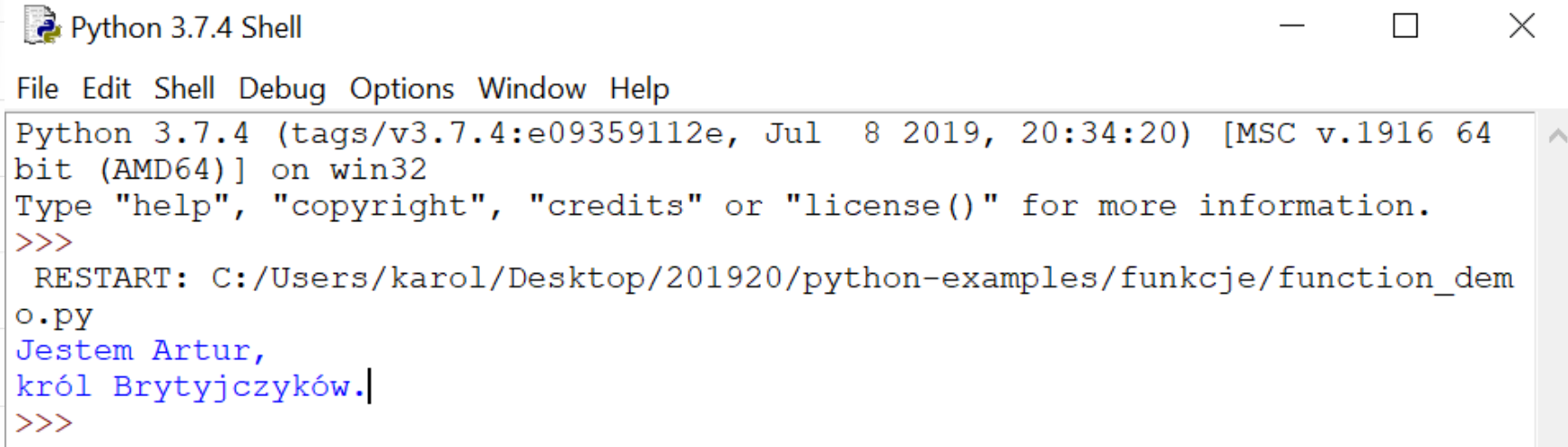
Definiowanie i wywoływanie



```
function_demo.py
File Edit Format Run Options Window Help
#Program pokazuje przykładową definicję funkcji i jej wywołanie

#Definicja o nazwie message()
def message():
    print('Jestem Artur,')
    print('król Brytyjczyków.')

#Wywołanie funkcji message()
message()
```



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/function_dem
o.py
Jestem Artur,
król Brytyjczyków.|
>>>
```


Funkcje - powtórzenie

Definiowanie i wywoływanie

```
two_functions.py
File Edit Format Run Options Window Help
#Ten program zawiera dwie funkcje.

#Definicja funkcji głównej - main()
def main():
    print('Mam dla Ciebie wiadomość.')
    message()
    print('Żegnaj')

#Definicja funkcji message()
def message():
    print('Jestem Artur,')
    print('król Brytyjczyków.')

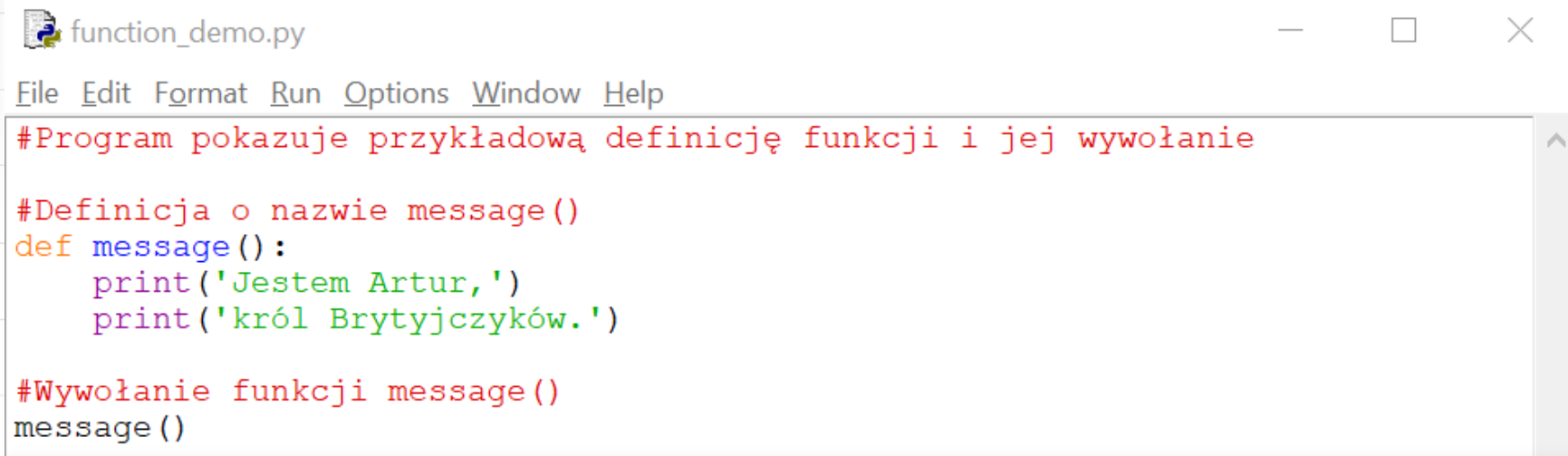
#Wywołanie funkcji głównej
main()
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/two_function
s.py
Mam dla Ciebie wiadomość.
Jestem Artur,
król Brytyjczyków.
Żegnaj
>>>
```

Funkcje - powtórzenie

Wcięcia

- Każdy wiersz bloku musi być wcięty
- Ostatni wcięty wiersz jest ostatnim blokiem kodu



```
function_demo.py
File Edit Format Run Options Window Help
#Program pokazuje przykładową definicję funkcji i jej wywołanie

#Definicja o nazwie message()
def message():
    print('Jestem Artur,')
    print('król Brytyjczyków.')

#Wywołanie funkcji message()
message()
```

Funkcje - powtórzenie

`main()`

- Tworzenie funkcji głównej nie jest wymogiem Pythona
- Tworzenie funkcji głównej jest konwencją, ale wykorzystywanie konwencji zwiększa czytelność kodu

Funkcje - powtórzenie

main ()



two_functions.py

File Edit Format Run Options Window Help

```
#Ten program zawiera dwie funkcje.

#Definicja funkcji głównej - main()
def main():
    print('Mam dla Ciebie wiadomość.')
    message()
    print('Żegnaj')


#Definicja funkcji message()
def message():
    print('Jestem Artur,')
    print('król Brytyjczyków.')

#Wywołanie funkcji głównej
main()
```



Funkcje - powtórzenie

Zmienne lokalne

 population.py

File Edit Format Run Options Window Help

```
#Program pokazuje, użycie dwóch zmiennych lokalnych
#o tych samych nazwach, w różnych funkcjach

def main():
    dolnoslaskie()
    mazowieckie()


#Funkcja dolnoslaskie() tworzy lokalną zmienną population.
def dolnoslaskie():
    population = 2902365
    print('Liczba mieszkańców województwa dolnośląskiego to ',\
          population, '.', sep='')

#Funkcja mazowieckie tworzy lokalną zmienną population.
def mazowieckie():
    population = 5391813
    print('Liczba mieszkańców województwa mazowieckiego to ',\
          population, '.', sep='')

main()
```

Funkcje - powtórzenie

Przekazywanie argumentów do funkcji

 pass_argument.py

File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie argumentu funkcji.
```

```
def main():  
    value = 13 #przypisanie zmiennej value wartości 13  
    #wywołanie funkcji show_double() z przekazaniem do niej argumentu  
    show_double(value)
```


```
#Funkcja show_double() pobiera argument  
#i wyświetla jego podwojoną wartość
```

```
def show_double(number):  
    result = number * 2  
    print(result)
```

```
main()
```

Funkcje - powtórzenie

Przekazywanie argumentów do funkcji

 pass_argument.py

File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie argumentu funkcji.
```

value



13

```
def main():
```

```
    value = 13 #przypisanie zmiennej value wartości 13
```

```
    #wywołanie funkcji show_double() z przekazaniem do niej argumentu
```

```
    show_double(value)
```

```
#Funkcja show_double() pobiera argument
```

```
#i wyświetla jego podwojoną wartość
```

```
def show_double(number):
```


```
    result = number * 2
```

```
    print(result)
```

```
main()
```

Funkcje - powtórzenie

Przekazywanie argumentów do funkcji

 pass_argument.py

File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie argumentu funkcji.
```

```
def main():
```

```
    value = 13 #przypisanie zmiennej value wartości 13
```

```
    #wywołanie funkcji show_double() z przekazaniem do niej argumentu
```

```
    show_double(value)
```

```
#Funkcja show_double() pobiera argument
```

```
#i wyświetla jego podwojoną wartość
```

```
def show_double(number):
```

```
    result = number * 2
```

```
    print(result)
```

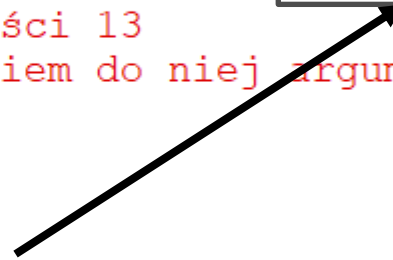
```
main()
```

value




13

number



Funkcje - powtórzenie

Przekazywanie argumentów do funkcji

 pass_argument.py

File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie argumentu funkcji.
```

```
def main():
```

```
    value = 13 #przypisanie zmiennej value wartości 13
```

```
    #wywołanie funkcji show_double() z przekazaniem do niej argumentu
```

```
    show_double(value)
```

```
#Funkcja show_double() pobiera argument
```

```
#i wyświetla jego podwojoną wartość
```

```
def show_double(number):
```

```
    result = number * 2
```

```
    print(result)
```

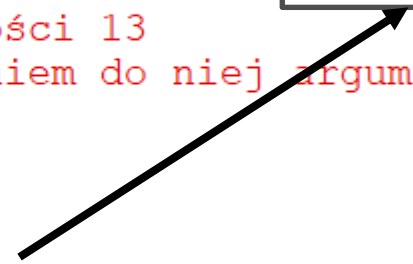
```
main()
```

value



13

number




result



26

Funkcje - powtórzenie

Przekazywanie argumentów do funkcji

 multiple_arguments.py

File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie dwóch argumentów do funkcji.
```

```
def main():  
    print('Suma liczb 12 i 45 wynosi')  
    #Wartości wyrażeń użytych w miejscu wywołania  
    #zostaną nadane zmiennym w funkcji show_sum().  
    show_sum(12,45)
```


```
#Funkcja show_sum() pobiera dwa argumenty  
#i wyświetla ich sumę.  
#W nawiasie umieszczono listę argumentów.  
#Nazwy argumentów są rozdzielone przecinkiem.
```

```
def show_sum(num1, num2):  
    result = num1 + num2  
    print(result)
```

```
main()
```

Funkcje - powtórzenie

Przekazywanie argumentów do funkcji

 multiple_arguments.py

File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie dwóch argumentów do funkcji.
```

```
def main():  
    print('Suma liczb 12 i 45 wynosi')  
    #Wartości wyrażeń użytych w miejscu wywołania  
    #zostaną nadane zmiennym w funkcji show_sum().  
    show_sum(12,45)
```

```
#Funkcja show_sum() pobiera dwa argumenty  
#i wyświetla ich sumę.  
#W nawiasie umieszczono listę argumentów.  
#Nazwy argumentów są rozdzielone przecinkiem.
```

```
def show_sum(num1, num2):  
    result = num1 + num2  
    print(result)
```

```
main()
```

num1



12

num2



45

Funkcje - powtórzenie

Zmiana argumentu funkcji

```
change_me.py
File Edit Format Run Options Window Help
#Program pokazuje, co się stanie po zmianie
#wartości argumentu w funkcji.

def main():
    value = 99
    print('Wartość wynosi',value)
    change_me(value)
    print('Wartość w funkcji main() wynosi',value)

def change_me(arg):
    print('Do funkcji change_me() przekazano wartość',arg)
    arg = 0
    print('Przypisano nową wartość do zmiennej arg')
    print('Teraz wartość wynosi',arg)

main()
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/change_me.py
Wartość wynosi 99
Do funkcji change_me() przekazano wartość 99
Przypisano nową wartość do zmiennej arg
Teraz wartość wynosi 0
Wartość w funkcji main() wynosi 99
>>>
```

Funkcje - powtórzenie

Zmiana argumentu funkcji

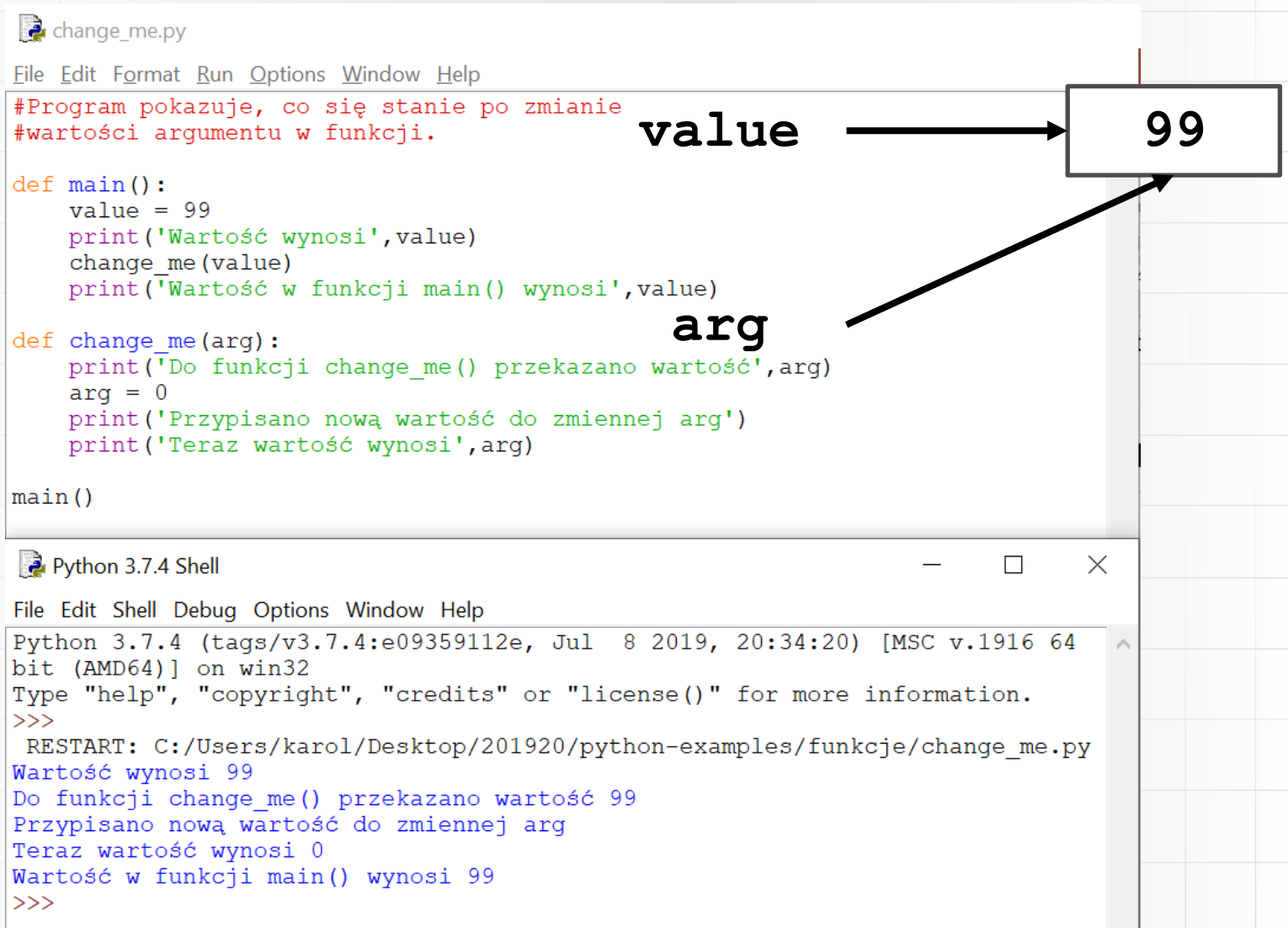
```
change_me.py
File Edit Format Run Options Window Help
#Program pokazuje, co się stanie po zmianie
#wartości argumentu w funkcji.
def main():
    value = 99
    print('Wartość wynosi',value)
    change_me(value)
    print('Wartość w funkcji main() wynosi',value)
def change_me(arg):
    print('Do funkcji change_me() przekazano wartość',arg)
    arg = 0
    print('Przypisano nową wartość do zmiennej arg')
    print('Teraz wartość wynosi',arg)
main()
```

value → **99**

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/change_me.py
Wartość wynosi 99
Do funkcji change_me() przekazano wartość 99
Przypisano nową wartość do zmiennej arg
Teraz wartość wynosi 0
Wartość w funkcji main() wynosi 99
>>>
```

Funkcje - powtórzenie

Zmiana argumentu funkcji



```
change_me.py
File Edit Format Run Options Window Help
#Program pokazuje, co się stanie po zmianie
#wartości argumentu w funkcji.

def main():
    value = 99
    print('Wartość wynosi',value)
    change_me(value)
    print('Wartość w funkcji main() wynosi',value)

def change_me(arg):
    print('Do funkcji change_me() przekazano wartość',arg)
    arg = 0
    print('Przypisano nową wartość do zmiennej arg')
    print('Teraz wartość wynosi',arg)

main()
```

value → **99**

arg → **99**

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/change_me.py
Wartość wynosi 99
Do funkcji change_me() przekazano wartość 99
Przypisano nową wartość do zmiennej arg
Teraz wartość wynosi 0
Wartość w funkcji main() wynosi 99
>>>
```

Funkcje - powtórzenie

Zmiana argumentu funkcji

```
change_me.py
File Edit Format Run Options Window Help
#Program pokazuje, co się stanie po zmianie
#wartości argumentu w funkcji.
def main():
    value = 99
    print('Wartość wynosi',value)
    change_me(value)
    print('Wartość w funkcji main() wynosi',value)
def change_me(arg):
    print('Do funkcji change_me() przekazano wartość',arg)
    arg = 0
    print('Przypisano nową wartość do zmiennej arg')
    print('Teraz wartość wynosi',arg)
main()
```

value → 99

arg → 0

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/change_me.py
Wartość wynosi 99
Do funkcji change_me() przekazano wartość 99
Przypisano nową wartość do zmiennej arg
Teraz wartość wynosi 0
Wartość w funkcji main() wynosi 99
>>>
```

Funkcje - powtórzenie

Argumenty nazwane

show_interest_01.py



File Edit Format Run Options Window Help

```
#Ten program pokazuje przykład wywołania funkcji
#z użyciem argumentów nazwanych.

#Funkcja główna wywołuje funkcję wyświetlającą obliczony wynik.
def main():
    #W tym wywołaniu argumenty przyjmują wartości na podstawie pozycji.
    show_interest(1000,0.03,3)

    #W tym wywołaniu argumenty przyjmują wartości zgodnie z nazwą.
    #Kolejność nie ma znaczenia.
    show_interest(rate = 0.03, periods = 3, deposit = 1000)

#Funkcja wyświetlająca wysokość odsetek należnych od zdeponowanej kwoty
#w zależności od czasu trwania depozytu oraz wysokości oprocentowania.
def show_interest(deposit, rate, periods):
    interest = deposit * rate * periods
    print('Wysokość odsetek wynosi',
          format(interest, '.2f')
        )

#Wywołanie funkcji głównej
main()
```


Funkcje - powtórzenie

Funkcje zwracające wartość

- Ogólna postać funkcji zwracającej wartość

```
def nazwa_funkcji() :  
    polecenie  
    polecenie  
    polecenie  
    itd.  
    return wyrażenie
```

Funkcje - powtórzenie

Funkcje zwracające wartość

```
return_result_01.py
File Edit Format Run Options Window Help
#Program pokazuje definicję i wywołanie funkcji zwracającej wartość.
#Program pyta użytkownika o liczbę jabłek i pomarańczy.
#Następnie wywołuje funkcję sum() do obliczenia łącznej liczby owoców
#i wyświetla stosowny komunikat.
def main():
    apples = int(input('Ile masz jabłek? '))
    oranges = int(input('Ile masz pomarańczy? '))
    print('Liczba wszystkich owoców ',sum(apples,oranges),',.',sep='')

#Funkcja sum oblicza sumę dwóch liczb
def sum(num1, num2):
    result = num1 + num2 #zapisanie wyniku w zmiennej pomocniczej
    return result      #zwrócenie wyniku

main()
```

Funkcje - powtórzenie

Funkcje zwracające wartość

return_result_02.py



File Edit Format Run Options Window Help

```
#Program pokazuje definicję i wywołanie funkcji zwracającej wartość.  
  
#Program pyta użytkownika o liczbę jabłek i pomarańczy.  
#Następnie wywołuje funkcję sum() do obliczenia łącznej liczby owoców  
#i wyświetla stosowny komunikat.  
def main():  
    apples = int(input('Ile masz jabłek? '))  
    oranges = int(input('Ile masz pomarańczy? '))  
    print('Liczba wszystkich owoców ', sum(apples, oranges), '.', sep='')  
  
#Funkcja sum oblicza sumę dwóch liczb  
def sum(num1, num2):  
    return num1 + num2 #polecenie return może zwracać wartość wyrażenia  
  
main()
```



Funkcje rekurencyjne

- Rekurencją nazywamy odwoływanie się funkcji do samej siebie
- Rekurencję można wykorzystać do rozwiązywania problemów, które można podzielić na mniejsze wersje tego samego problemu



Funkcje rekurencyjne

recursion_endless.py



File Edit Format Run Options Window Help

```
#Program pokazuje przykładową funkcję rekurencyjną.
```

```
#Jest to przykład nieskończonej rekurencji.
```

```
def main():  
    message()
```

```
#Funkcja message() wywołuje samą siebie.
```

```
def message():  
    print('To jest funkcja rekurencyjna')  
    message()
```

```
main()
```



Funkcje rekurencyjne

- Istotnym zagadnieniem przy wykorzystaniu rekurencji jest warunek stopu



Funkcje rekurencyjne

recursion_01.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykładową funkcję rekurencyjną.
```

```
def main():  
    message(5)
```

```
#I :liczba całkowita określająca liczbę  
#   wywołań rekurencyjnych  
#P :jeśli liczba wywołań jest dodatnia  
#   wyświetl informację  
#   wywołaj rekurencyjnie funkcję message()  
#   z liczbą wywołań mniejszą o jeden  
#0 :brak
```

```
def message(times):  
    if times > 0:  
        print('To jest funkcja rekurencyjna')  
        message(times - 1)
```

```
main()
```



Funkcje rekurencyjne

recursion_01.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykładową funkcję rekurencyjną.
```

```
def main():  
    message(5)
```

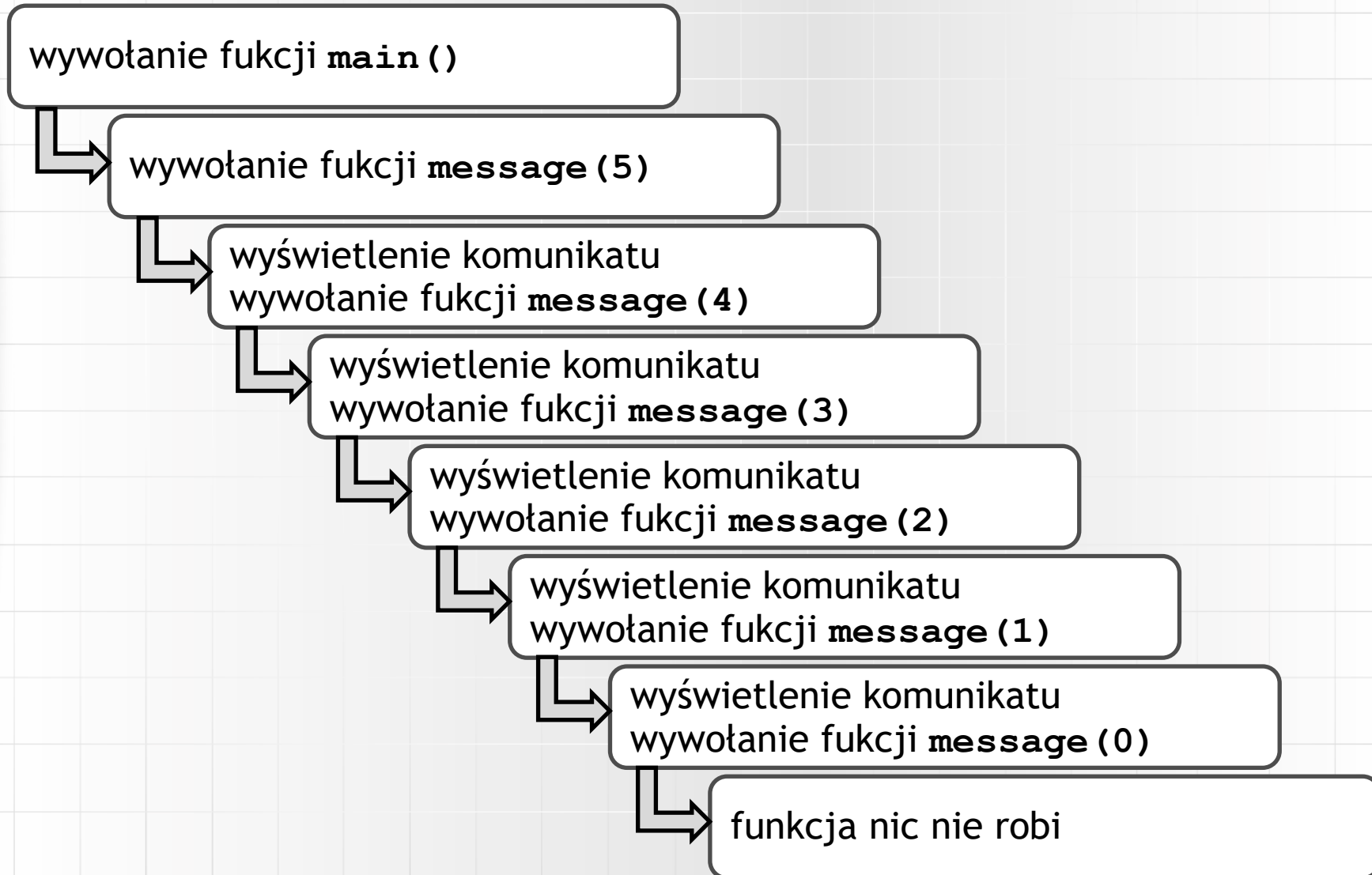
```
To jest funkcja rekurencyjna  
To jest funkcja rekurencyjna  
To jest funkcja rekurencyjna  
To jest funkcja rekurencyjna  
To jest funkcja rekurencyjna  
>>>
```

```
def message(times):  
    if times > 0:  
        print('To jest funkcja rekurencyjna')  
        message(times - 1)
```

```
main()
```




Funkcje rekurencyjne





Funkcje rekurencyjne

recursion_02.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykładową funkcję rekurencyjną.
```

```
def main():  
    message(5)
```

```
#I :liczba całkowita określająca liczbę  
#   wywołań rekurencyjnych  
#P :jeśli liczba wywołań jest dodatnia  
#   wyświetl informację  
#   wywołaj rekurencyjnie funkcję message()  
#   z liczbę wywołań mniejszą o jeden  
#O :brak
```

```
def message(times):  
    if times > 0:  
        print('Wartość parametru times', times)  
        message(times - 1)
```

```
main()
```



Funkcje rekurencyjne

recursion_02.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykładową funkcję rekurencyjną.
```

```
def main():  
    message(5)
```

```
Wartość parametru times 5  
Wartość parametru times 4  
Wartość parametru times 3  
Wartość parametru times 2  
Wartość parametru times 1  
>>>
```

```
def message(times):  
    if times > 0:  
        print('Wartość parametru times', times)  
        message(times - 1)
```

```
main()
```



Funkcje rekurencyjne

recursion_03.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykładową funkcję rekurencyjną.
```

```
def main():  
    message(5)
```

```
#I :liczba całkowita określająca liczbę wywołań  
#   rekurencyjnych
```

```
#P :jeśli liczba wywołań jest dodatnia
```

```
#     wywołaj rekurencyjnie funkcję message()
```

```
#     z liczbę wywołań mniejszą o jeden
```

```
#     wyświetl informację
```

```
#0 :brak
```

```
def message(times):
```

```
    if times > 0:
```

```
        #zmieniona kolejność instrukcji
```

```
        message(times - 1)
```

```
        print('Wartość parametru times', times)
```

```
main()
```



Funkcje rekurencyjne

recursion_03.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykładową funkcję rekurencyjną.
```

```
def main():  
    message(5)
```

```
Wartość parametru times 1  
Wartość parametru times 2  
Wartość parametru times 3  
Wartość parametru times 4  
Wartość parametru times 5  
>>>
```

```
if times > 0:  
    #zmieniona kolejność instrukcji  
    message(times - 1)  
    print('Wartość parametru times', times)
```

```
main()
```

Funkcje rekurencyjne

Silnia

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$$

$$n! = \begin{cases} 1 & \text{dla } n = 0, \\ n \cdot (n - 1)! & \text{dla } n > 0. \end{cases}$$

Funkcje rekurencyjne

Silnia

```
factorial.py
File Edit Format Run Options Window Help
#Program demonstruje dzialanie funkcji factorial
#obliczajacej wartosc silni rekurencyjnie

def main():
    print('5! =',factorial(5))

#I :liczba calkowita n
#P :oblicz silnie liczby n jako iloczyn
#   n * silnia (n-1) (jesli n jest rozne od zera)
#   dla zera - warunek stopu
#   0! = 1
#O :silnia liczby n

def factorial(n):
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)

main()
```



Algorytmy rekurencyjne

- Algorytm Euklidesa wyznaczania największego wspólnego dzielnika
- Potęgowanie rekurencyjne
- Wyznaczania elementów ciągu Fibonacciego



Algorytm Euklidesa

zakładając, że $m \leq n$

$$\text{NWD}(m, n) = \begin{cases} n, & \text{dla } m = 0, \\ \text{NWD}(n \bmod m, m), & \text{dla } m > 0. \end{cases}$$



Obliczanie potęgi

$$x^m = \begin{cases} x, & \text{dla } m = 1 \\ \left(x^{m/2}\right)^2, & m - \text{parzyste} \\ \left(x^{(m-1)/2}\right)^2 x, & m - \text{nieparzyste} \end{cases}$$

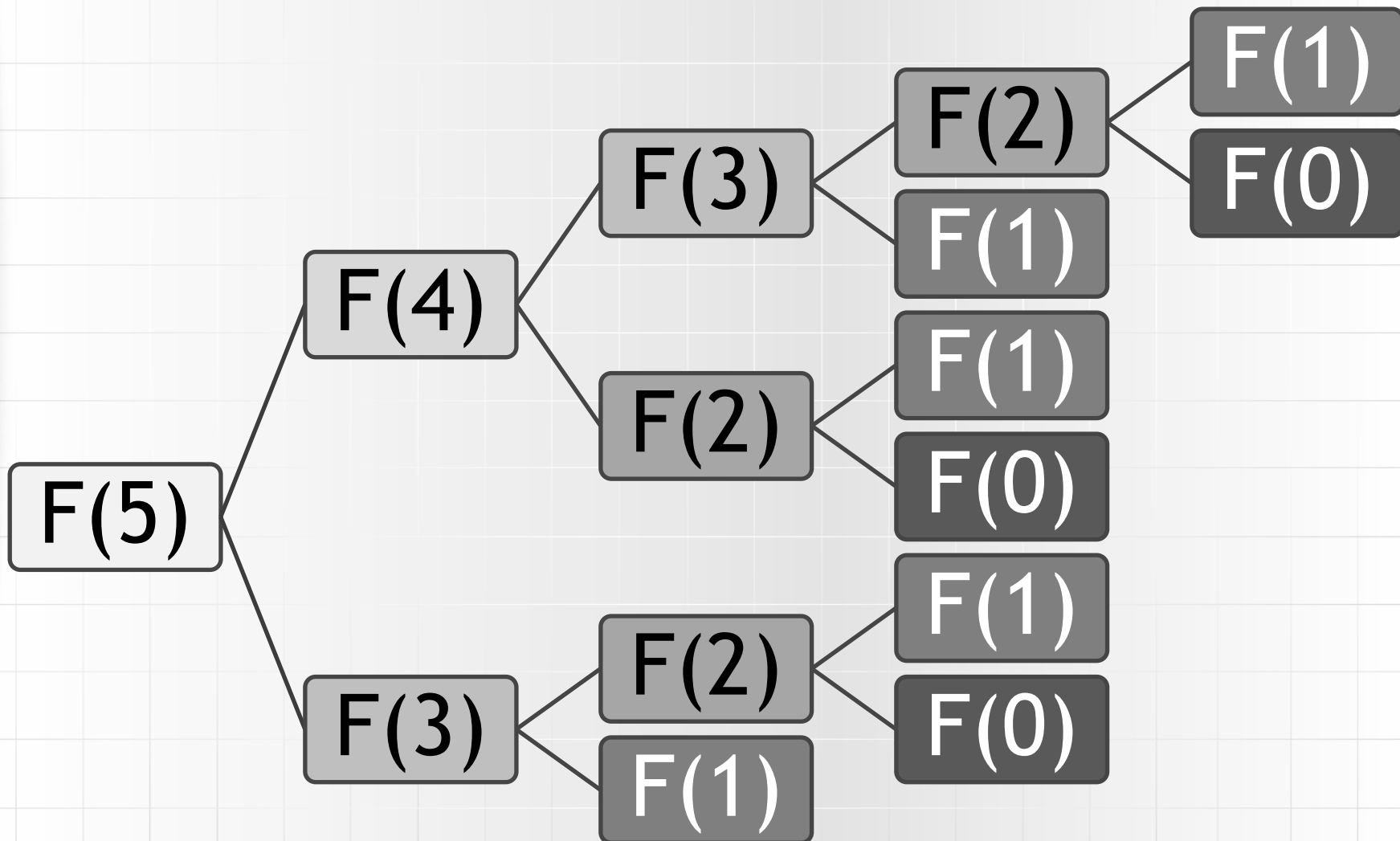


Ciąg Fibonacciego

$$F(n) = \begin{cases} 0 & \text{dla } n = 0, \\ 1 & \text{dla } n = 1, \\ F(n-1) + F(n-2) & \text{dla } n > 1. \end{cases}$$

Ciąg Fibonacciego

Wywołania rekurencyjne





Podsumowanie

- Co to jest algorytm?
- Co to jest program?
- Funkcje - powtórzenie
- Funkcje rekurencyjne