Politechnika Wrocławska

Podstawy programowania W11OPA-SI0072G
Wstęp do programowania W11FTE-SI0141WL
Wstęp do programowania W11IKW-SI0080WL

rok akademicki 2024/25
semestr letni

Wykład 2

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 221

Politechnika Wrocławska

Plan prezentacji (1)

- Algorytmy
  - Algorytmy i programy
  - Algorytmy liniowe
  - Algorytmy z rozgałęzieniami

Politechnika Wrocławska

Plan prezentacji (2)

- Podstawowe funkcje i elementy języka python
  - Funkcja print()
  - Ciągi tekstowe
  - Komentarze
  - Zmienne
  - Literały liczbowe
  - Funkcja input()
  - Funkcje
  - Operatory matematyczne
  - Instrukcja warunkowa i operatory logiczne

Politechnika Wrocławska

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

M. M. Sysło, Algorytmy, WSiP, 2002

Politechnika Wrocławska

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://commons.wikimedia.org/wiki/File:Al_Khwarizmi%27s_Monument_in_Khiva.png

https://commons.wikimedia.org/wiki/Category:Muhammad_ibn_Musa_al-Khwarizmi#/media/File:Image-Al-Kitāb_al-muḫtaṣar_fī_ḥisāb_al-ğabr_wa-l-muqābala.jpg

M. M. Sysło, Algorytmy, WSiP, 2002

Politechnika Wrocławska

Przykład algorytmu

KAKAOWE CIASTO BEZ PIECZENIA

SZYBKIE CIASTO | WIGILIA I ŚWIĘTA | WIELKANOC | DESERY BEZ PIECZENIA | CIASTA | CIASTA BEZ PIECZENIA

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).

https://www.kwestiasmaku.com/przepis/kakaowe-ciasto-bez-pieczenia

Politechnika Wrocławska

Przykład algorytmu

SKŁADNIKI

KREM CZEKOLADOWY

- 1 litr mleka
- 3 pełne łyżki mąki pszennej
- 3 pełne łyżki mąki ziemniaczanej
- 3 łyżki kakao
- 1 szklanka cukru
- 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

- Odlać 1 i 1/2 szklanki mleka i dokładnie wymieszać je (np. rózgą) z mąką pszenną, ziemniaczaną, kakao, cukrem, żółtkami oraz likierami jeśli ich używamy.
- Resztę mleka zagotować (dokładnie, aż zacznie kipieć), następnie wlewać do niego mieszankę mleka, mąki i żółtek, jednocześnie energicznie mieszając rózgą. 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.

https://www.kwestiasmaku.com/przepis/kakaowe-ciasto-bez-pieczenia

Politechnika Wrocławska

Przykład algorytmu

SKŁADNIKI

KREM CZEKOLADOWY

- 1 litr mleka
- 3 pełne łyżki mąki pszennej
- 3 pełne łyżki mąki ziemniaczanej
- 3 łyżki kakao
- 1 szklanka cukru
- 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

- Odlać 1 i 1/2 szklanki mleka i dokładnie wymieszać je (np
pszenną, ziemniaczaną, kakao, cukrem, żółtkami oraz lik
używamy.
- Resztę mleka zagotować (dokładnie, aż zacznie kip
wlewać do niego mieszankę mleka, mąki i żółtek, jednocześ
mieszając rózgą. Zagotować co chwilę mieszając.
- Po zagotowaniu gotowy budyń odstawić z ognia, prz
miski i całkowicie ostudzić (na wierzch można położyć folię
nie zrobił się kożuch).
- Miękkie masło ubijać przez ok. 3 minuty aż się napuszy, następnie stopniowo, w krótkich odstępach czasu, dodawać budyń ciągle ubijając.

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, z tym, że druga warstwa od dołu ma mieć zamiast kremu - dżem. Na wierzchu też ma być cienka warstwa kremu.
- Ciasto oprószyć kakao i wstawić do lodówki na kilka godzin lub na całą noc. Ciasto można przygotować dzień wcześniej.

https://www.kwestiasmaku.com/przepis/kakaowe-ciasto-bez-pieczenia

Politechnika Wrocławska

Przykład algorytmu

SKŁADNIKI

KREM CZEKOLADOWY

- 1 litr mleka
- 3 pełne łyżki mąki pszennej
- 3 pełne łyżki mąki ziemniaczanej
- 3 łyżki kakao
- 1 szklanka cukru
- 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

- Odlać 1 i 1/2 szklanki mleka i dokładnie wymieszać je (np
pszenną, ziemniaczaną, kakao, cukrem, żółtkami oraz lik
używamy.
- Resztę mleka zagotować (dokładnie, aż zacznie kip
wlewać do niego mieszankę mleka, mąki i żółtek, jednoczes
mieszając rózgą. Zagotować co chwilę mieszając.
- Po zagotowaniu gotowy budyń odstawić z ognia, prz
miski i całkowicie ostudzić (na wierzch można położyć folię
nie zrobił się kożuch).
- Miękkie masło ubijać przez ok. 3 minuty aż się napuszy, następnie stopniowo, w krótkich odstępach czasu, dodawać budyń ciągle ubijając.

- Formę o wyn
masłem i wyłoży
herbatniki i kren
druga warstwa c
ma być cienka w
- Ciasto oprósz
noc. Ciasto możn

https://www.kwestiasmaku.com/przepis/kakaowe-ciasto-bez-pieczenia

Algorytm Euklidesa

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

Politechnika Wrocławska

Co to jest program?

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

Politechnika Wrocławska

Program i przepis a algorytm

Gotowanie

- Składniki
- Przepis
- Kucharz i kuchnia
- Danie

Informatyka

- Dane
- Program
- Komputer
- Wynik

Politechnika Wrocławska

Reprezentacje problemów

- Problemy i algorytmy mogą mieć różnorodne reprezentacje:
  - opis słowny
  - lista kroków
  - schemat blokowy

Politechnika Wrocławska

Reprezentacje problemów

- Oblicz wartość funkcji $$f(x) = \frac{x}{|x|}$$

- Konieczne jest sformułowanie specyfikacji, która określa warunki jakie powinny spełniać dane wejściowe, oraz jaki jest związek wyników z danymi.

- Dane: dowolna liczba rzeczywista x.
- Wynik: wartość funkcji $f(x)$, jeśli x jest różne od zera i zero w przeciwnym przypadku

Politechnika Wrocławska

Reprezentacje problemów

Opis słowny

- Dane: dowolna liczba rzeczywista x.
- Wynik: wartość funkcji $f(x)$, określonej wzorem

$$f(x) = \begin{cases} -1 & \text{dla } x < 0, \\ 0 & \text{dla } x = 0, \\ +1 & \text{dla } x > 0. \end{cases}$$

Politechnika Wrocławska

Reprezentacje problemów

Lista kroków

- Dane: dowolna liczba rzeczywista x.
- Wynik: wartość funkcji f(x), określonej wzorem

Krok 0. Wczytaj wartość danej x.

Krok 1. Jeśli x > 0, wynikiem jest 1. Zakończ algorytm.

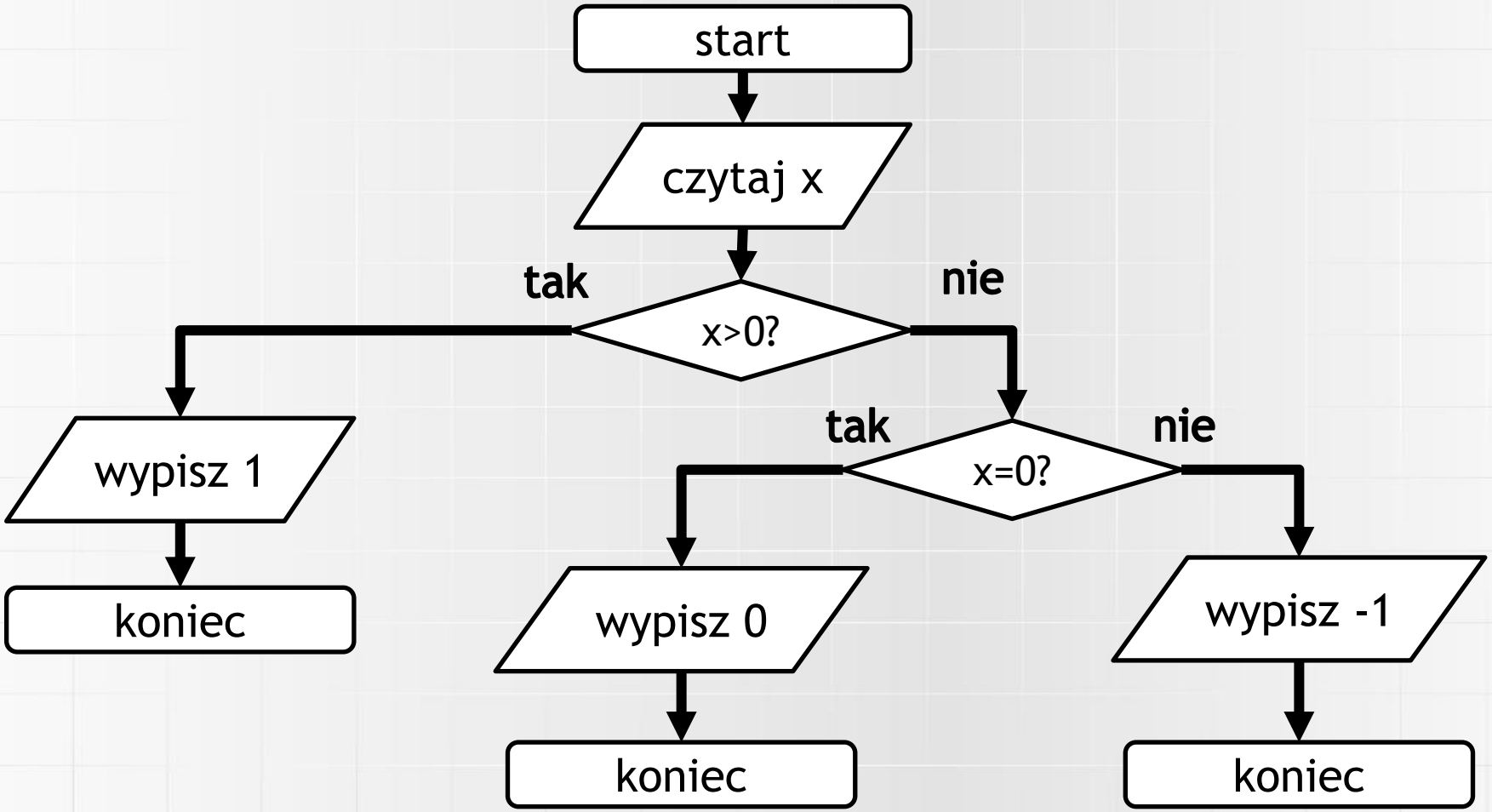
Krok 2. {*W tym przypadku x ≤ 0*} Jeśli x = 0, wynikiem jest 0. Zakończ algorytm.

Krok 3. {*W tym przypadku x < 0*} Wynikiem jest -1. Zakończ algorytm.

Politechnika Wrocławska

Reprezentacje problemów

Schemat blokowy



Politechnika Wrocławska

Algorytmy liniowe

- Algorytm liniowy ma postać listy kroków, które mają być wszystkie wykonane zgodnie z kolejnością w jakiej występują
- Nawet w przypadku prostych algorytmów warto dbać o efektywność wykonywania obliczeń

Politechnika Wrocławska

Algorytmy liniowe

- Przykład: obliczanie wartości wielomianu
- Wykorzystanie schematu Hornera pozwala zredukować liczbę mnożeń – tym większy zysk im wyższy stopień wielomianu

$$w(x) = ax^2 + bx + c$$
$$w(x) = (ax + b)x + c$$

$$v(x) = ax^3 + bx^2 + cx + d$$
$$v(x) = ((ax + b)x + c)x + d$$

Politechnika Wrocławska

Algorytmy z rozgałęzieniami

- Najczęściej przebieg algorytmu uzależniony jest od spełnienia pewnych warunków – wykorzystanie instrukcji warunkowej

Politechnika Wrocławska

Funkcja print()

- **Funkcja** to przygotowany fragment kodu przeprowadzający pewną operację
- Uruchomienie funkcji nazywa się **wywołaniem**
- W nawiasach umieszcza się **argument**

- Przykładowo wywołanie funkcji:

```
print('Hello world!')
```

spowoduje wypisanie napisu na ekran.

Politechnika Wrocławska

Funkcja print()

Przykłady

print01.py print02.py print03.py print04.py print0

```
1  print('Tony Gaddis')
2  print('Python dla zupełnie początkujących')
3  print('Helion, 2019')
```

print01.py print02.py print03.py print04.py print05.p

```
1  print("Tony Gaddis")
2  print("Python dla zupełnie początkujących")
3  print("Helion, 2019")
```

Politechnika Wrocławska

Funkcja print()

Przykłady

print_apostrophe.py | print_multiline.py

```
1  print("Znak apostrofu ' można umieści, jeśli literał znakowy jest ujęty w cudzysłów.")
2  print('Tak wygląda znak cudzysłowu: " .')
3  print("""W tym łańcuch znakowym mamy ' i " """)
4  print('''oraz w tym: ' i " ''')
5
```

print_apostrophe.py | print_multiline.py

```
1  print("""Jeden
2  Dwa
3  Trzy""")
4
```

Politechnika Wrocławska

Ciągi tekstowe

- W programach będą się pojawiały dane różnych typów
- Napis **`Hello world!`** w przykładowym wywołaniu funkcji

  ```
  print('Hello world!')
  ```

  jest **ciągiem tekstowym**.
- Gdy ciąg tekstowy pojawia się w kodzie programu nazywamy go **literałem znakowym**.

Politechnika Wrocławska

Komentarze

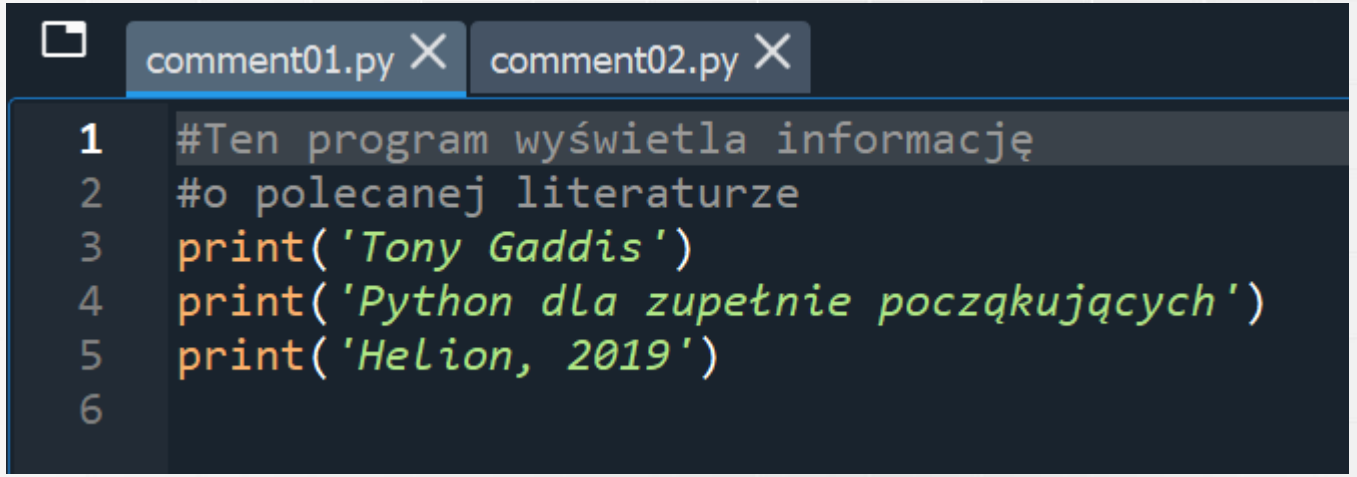
- Komentarz – informacja umieszczona w kodzie źródłowym programu, która objaśnia jego działanie
- Komentarze są ignorowane przez interpreter
- Komentarze ułatwiają zrozumienie kodu

Politechnika Wrocławska

Komentarze

Przykład 1

- Komentarz wyjaśnia przeznaczenie programu



Politechnika Wrocławska

Komentarze

Przykład 2

- Komentarz na końcu wiersza objaśnia jego działanie

comment01.py comment02.py

```
1  print('Tony Gaddis')                          #Wyświetlenie informacji o autorze
2  print('Python dla zupełnie początkujących')  #Wyświetlenie tytułu
3  print('Helion, 2019')                        #Wyświetlenie wydawcy i roku wydania
```

Komentarze

- W trakcie pracy nad programem można wykorzystywać komentarze do wyłączania pewnych fragmentów kodu

Politechnika Wrocławska

Zmienne

- **Zmienna** to miejsce w pamięci komputera reprezentowane przez określoną nazwę
- **Przypisanie** jest używane do utworzenia zmiennej i określenia jej jako odwołania do pewnego fragmentu danych

```
wiek = 19
```

Politechnika Wrocławska

Przypisanie

- Ogólna postać polecenia przypisania:

  `zmienna = wyrażenie`

- Znak równości jest **operatorem przypisania**

Politechnika Wrocławska

Zmienne

- Zmienne można przekazywać do funkcji
- Przykład wykorzystuje funkcję print()

variables.py

```
1  #Program pokazuje przykład użycia zmiennej.
2  sala = 249
3  print('Kurs "Wstęp do programowania" jest prowadzony w sali')
4  print(sala)
5
```

Console 1/A

```
Kurs "Wstęp do programowania" jest prowadzony w sali
249
```

Politechnika Wrocławska

Zmienne

- Zmiennej nie można używać przed przypisaniem jej wartości

variables.py | variables_error01.py

```
1  #Program pokazuje ZłY przykład użycia zmiennej.
2  print('Kurs "Wstęp do programowania" jest prowadzony w sali')
3  print(sala) #uzycie zmiennej przed przypisaniem jej wartości
4  sala = 249
5
```

Console 1/A

```
Kurs "Wstęp do programowania" jest prowadzony w sali
Traceback (most recent call last):

  File C:\ProgramData\anaconda3-2024\Lib\site-packages\spyder_kernels\py3compat.py:356 in compat_exec
    exec(code, globals, locals)

  File c:\dane karola\repozytoria\python-examples\zmienne\variables_error01.py:3
    print(sala) #uzycie zmiennej przed przypisaniem jej wartości

NameError: name 'sala' is not defined
```

Politechnika Wrocławska

Zmienne

- Należy uważać na literówki

variables.py | variables_error01.py | variables_error02.py

```
1  #Program pokazuje przykład BŁĘDU przy użyciu zmiennej.
2  sala = 249
3  print('Kurs "Wstęp do programowania" jest prowadzony w sali')
4  print(Sala) #błędna nazwa zmiennej - jest: 'Sala' - powinno być: 'sala'
5
```

Console 1/A

```
Kurs "Wstęp do programowania" jest prowadzony w sali
Traceback (most recent call last):

  File C:\ProgramData\anaconda3-2024\Lib\site-packages\spyder_kernels\py3compat.py:356 in compat_exec
    exec(code, globals, locals)

  File c:\dane karola\repozytoria\python-examples\zmienne\variables_error02.py:4
    print(Sala) #błędna nazwa zmiennej - jest: 'Sala' - powinno być: 'sala'

NameError: name 'Sala' is not defined
```

Politechnika Wrocławska

Zmienne

- Nazwą zmiennej nie mogą być słowa kluczowe Pythona
- Nazwa zmiennej nie może zawierać spacji
- Pierwszym znakiem musi być litera lub podkreślenie (_)
- Każdym kolejnym znakiem może być litera, cyfra lub podkreślenie
- Rozróżniana jest wielkość liter

Politechnika Wrocławska

Zmienne

- Nazwa zmiennej powinny wskazywać do czego służy dana zmienna
- Jeśli nazwa zmiennej ma się składać z kilku wyrazów to wygodnie jest je oddzielić znakiem podkreślenia

  `liczba_produktow` zamiast **`liczbaproduktow`**
- Inna możliwa konwencja, to rozpoczynanie kolejnych wyrazów wielką literą

  `liczbaProduktow`

Politechnika Wrocławska

Funkcja print()

Dwa argumenty wywołania

variables.py | variables01.py

```
1  #Program pokazuje przykład wywołania funkcji print z dwoma argumentami.
2  sala = 249
3  print('Kurs "Wstęp do programowania" jest prowadzony w sali', sala)
4
```

Console 1/A

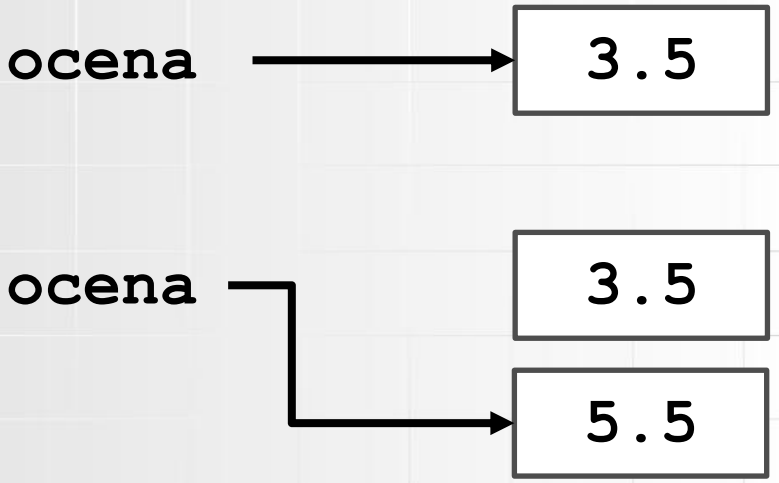
```
Kurs "Wstęp do programowania" jest prowadzony w sali 249
```

Zmienne

Ponowne przypisanie

variables.py | variables01.py | variables02.py

```
1  #Program pokazuje ponowne przypisanie zmiennej.
2
3  #Wartość jest przypisywana zmiennej ocena
4  ocena = 3.5
5  print('Moja ocena to:', ocena)
6
7  #Ponowne przypisanie zmiennej ocena
8  ocena = 5.5
9  print('Moja ocena jest wyższa! To:',ocena)
10
```



Politechnika Wrocławska

Literały i liczbowe typy danych

- Poszczególne typy liczb przechowywane są w odmienny sposób
- Typy danych służą do kategoryzowania wartości w pamięci
- Wartości z przykładów:

```
sala = 249   # liczba całkowita
             # typ int   (integer)
ocena = 3.5  # liczba rzeczywista
             # typ float (floating-point)
```

Politechnika Wrocławska

Literały i liczbowe typy danych

- Liczba zapisana w kodzie programu nazywa się **literałem liczbowym**
- Literał liczbowy zapisany w postaci liczby całkowitej jest uznawany za typ **`int`**

`249, -9, 7`

- Literał liczbowy zapisany w postaci liczby z częścią ułamkową jest uznawany za typ **`float`**

`1.5, 3.14, 5.0`

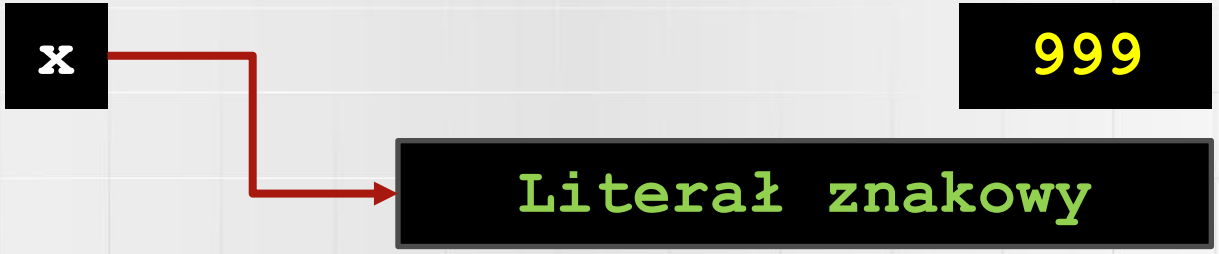
Politechnika Wrocławska

Przypisanie zmiennej wartości innego typu

- Zmienna w Pythonie może odwoływać się do wartości dowolnego typu

variables.py | variables01.py | variables02.py | variables03.py

```
1  #Program pokazuje przypisanie zmiennej wartości innego typu.
2
3  x = 999 # zmiennej x jest przypisana wartość typu całkowitego
4  print(x)
5
6  x = 'Literał, znakowy' # zmiennej x jest przypisana wartość typu str
7  print(x)
8
```



Politechnika Wrocławska

Odczyt danych wejściowych

- Do pobierania danych wejściowych dostarczanych za pomocą klawiatury służy funkcja **`input()`**
- Przed pobraniem danych należy poinformować użytkownika, jakie dane są potrzebne
- Ogólna postać wywołania

```
zmienna = input(ciąg tekstowy)
```

- Przykładowo

```
name = input('Jak masz na imię?')
```

Politechnika Wrocławska

Odczyt danych wejściowych

input_string.py

```
1  """
2  Program pokazuje pobieranie łańcuchów znakowych z klawiatury.
3  """
4
5  #Pobranie pseudonimu
6  nickname = input('Podaj pseudonim ')
7
8  #Wyświetlenie wiadomości powitalnej
9  print('Witaj,', nickname)
```

Console 1/A

```
Podaj pseudonim Karol
Witaj, Karol
```

Politechnika Wrocławska

Odczyt danych wejściowych

- Funkcja **`input()`** zawsze zwraca wprowadzone dane wejściowe jako ciąg tekstowy
- Aby zmienić typ danych (skonwertować dane) można wykorzystać funkcje **`int()`** oraz **`float()`**
- Przykładowo

```
n = int(input('Podaj liczbę całkowitą'))
x = float(input('Podaj liczbę rzeczywistą'))
```

Politechnika Wrocławska

Funkcje

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

Politechnika Wrocławska

Funkcje

Definiowanie i wywoływanie

- Podstawowy schemat definicji funkcji

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

nagłówek funkcji

blok poleceń

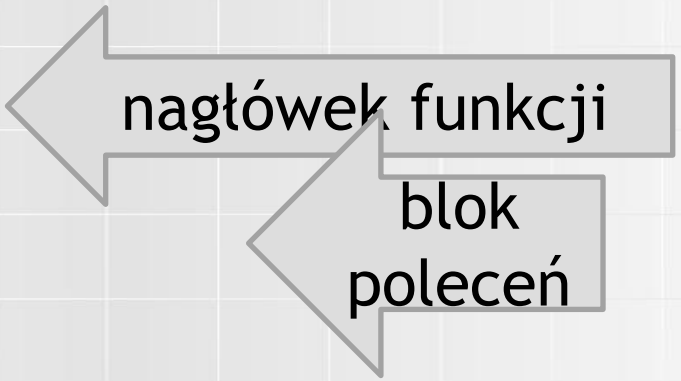
Politechnika Wrocławska

Funkcje

Definiowanie i wywoływanie

- Przykładowa definicja funkcji

```
def message():
    print('Jestem Artur,')
    print('król Brytyjczyków.')
```



- Wywołanie funkcji

```
message()
```

Politechnika Wrocławska

Funkcje

Definiowanie i wywoływanie

function_demo.py

```
1  #Program pokazuje przykładową definicję funkcji i jej wywołanie
2
3  #Definicja o nazwie message()
4  def message():
5      print('Jestem Artur,')
6      print('król Brytyjczyków.')
7
8  #Wywołanie funkcji message()
9  message()
10
```

Console 1/A

```
Jestem Artur,
król Brytyjczyków.
```

Politechnika Wrocławska

Funkcje

Definiowanie i wywoływanie

two_functions.py

```
1  #Ten program zawiera dwie funkcje.
2
3  #Definicja funkcji głównej - main()
4  def main():
5      print('Mam dla Ciebie wiadomość.')
6      message()
7      print('Żegnaj')
8
9  #Definicja funkcji message()
10 def message():
11     print('Jestem Artur,')
12     print('król Brytyjczyków.')
13
14 #Wywołanie funkcji głównej
15 main()
16
```

Console 1/A

```
Mam dla Ciebie wiadomość.
Jestem Artur,
król Brytyjczyków.
Żegnaj
```

Politechnika Wrocławska

Funkcje

Wcięcia

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

function_demo.py

```
1  #Program pokazuje przykładową definicję funkcji i jej wywołanie
2
3  #Definicja o nazwie message()
4  def message():
5      print('Jestem Artur,')
6      print('król Brytyjczyków.')
7
8  #Wywołanie funkcji message()
9  message()
10
```

Politechnika Wrocławska

Funkcje

Projektowanie programu

Projektowanie od ogółu do szczegółu

- Zadanie główne jakie ma wywołać program, dzielimy na mniejsze podzadania
- Każde podzadanie analizujemy i sprawdzamy, czy nie da się podzielić na mniejsze. Kontynuujemy dzielenie na podzadania, tak długo jak możemy
- Tworzymy kod dla każdego podzadania

Politechnika Wrocławska

Funkcje

Przykład

elephant.py

```
#Ten program wyświetla instrukcję krok po kroku
#jak schować słonia do lodówki.

#Funkcja main() zawiera logikę główną programu
def main():
    #Wyświetlenie komunikatu początkowego
    startup_message()
    input('Naciśnij Enter, aby przejść do kroku 1.')
    #Wyświetlenie instrukcji dla kroku 1.
    step1()
    input('Naciśnij Enter, aby przejść do kroku 2.')
    #Wyświetlenie instrukcji dla kroku 2.
    step2()
    input('Naciśnij Enter, aby przejść do kroku 3.')
    #Wyświetlenie instrukcji dla kroku 3.
    step3()
    input('Naciśnij Enter, aby zakończyć.')
    final_message()

```

Politechnika Wrocławska

Funkcje

Przykład

elephant.py

```
21  #Funkcja startup_message() wyświetla na ekranie
22  #komunikat początkowy.
23  def startup_message():
24      print('Ten program przeprowadzi Cię,')
25      print('przez proces chowania słonia do lodówki.')
26      print('Proces składa się z trzech prostych kroków.')
27      print()
28
29  #Funkcja step1() wyświetla instrukcję dla kroku 1.
30  def step1():
31      print('Krok 1.')
32      print('Otwórz lodówkę.')
33      print()
34
35  #Funkcja step2() wyświetla instrukcję dla kroku 2.
36  def step2():
37      print('Krok 2.')
38      print('Włóż słonia do lodówki.')
39      print()
```

Politechnika Wrocławska

Funkcje

Przykład

elephant.py

```
41  #Funkcja step3() wyświetla instrukcję dla kroku 3.
42  def step3():
43      print('Krok 3.')
44      print('Zamknij lodówkę.')
45      print()
46
47  #Funkcja final_message() wyświetla na ekranie
48  #komunikat końcowy.
49  def final_message():
50      print('\nJeśli postępowałeś zgodnie z instrukcjami,')
51      print('to słoń jest w lodówce.\n')
52
53  #Wywołanie funkcji main() i rozpoczęcie działania programu
54  main()
55
```

Politechnika Wrocławska

Funkcje

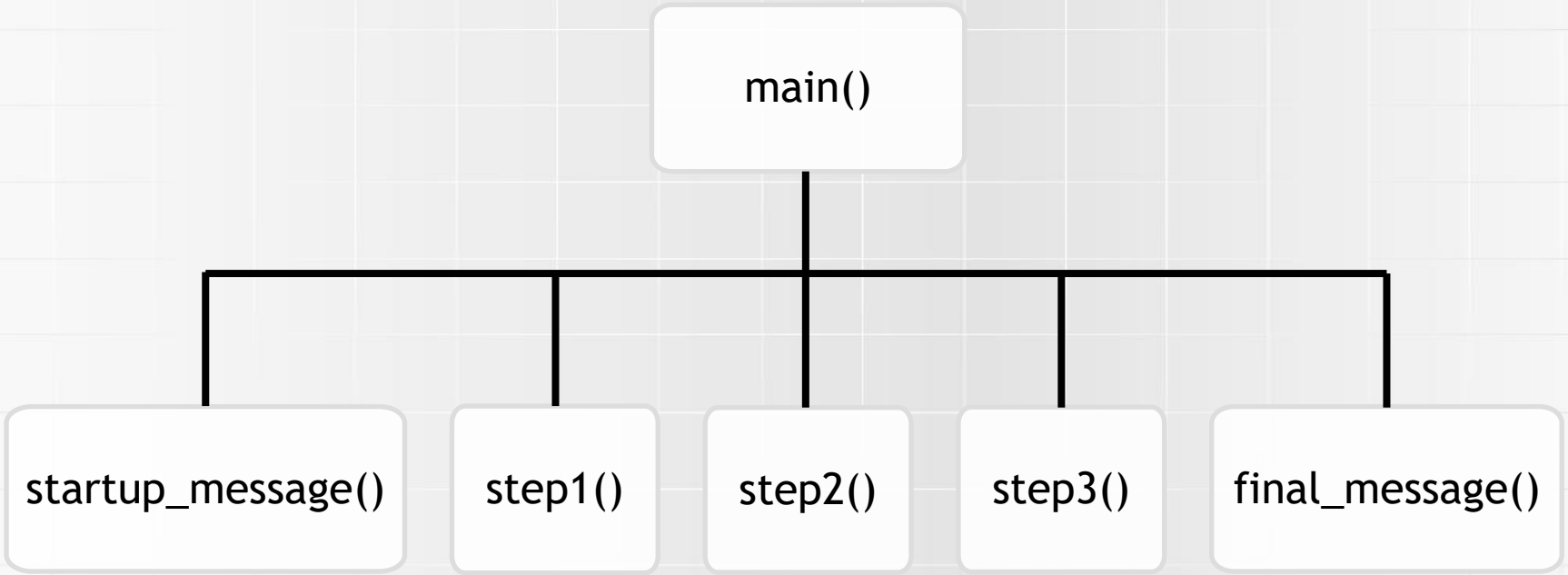
`main()`

- Python nie wyróżnia funkcji głównej
- Część programistów przyjmuje konwencję, w której tworzy się funkcję **`main()`**, ale nie jest to wymogiem języka

Politechnika Wrocławska

Funkcje

Schemat hierarchiczny - przykład



Politechnika Wrocławska

Funkcje

Zmienne lokalne

bad_local.py

```
1  #Program pokazuje BŁĘDNE użycie zmiennej lokalnej
2
3  #Defincja funkcji main()
4  def main():
5      get_name()
6      print('Witaj,', name) #Ten wiersz powoduje błąd.
7
8  #Funkcja get_name() pyta użytkownika o imię
9  def get_name():
10     name = input('Jak masz na imię? ')
11
12 #Wywołanie funkcji głównej
13 main()
```

- Zmienna **`name`** ma przypisaną wartość w funkcji **`get_name()`**
- Zmienna **`name`** jest niedostępna w funkcji **`main()`**

Politechnika Wrocławska

Funkcje

Zmienne lokalne

population.py

```
1  #Program pokazuje, użycie dwóch zmiennych lokalnych
2  #o tych samych nazwach, w różnych funkcjach
3
4  def main():
5      dolnoslaskie()
6      mazowieckie()
7
8  #Funkcja dolnoslaskie() tworzy lokalną zmienną population.
9  def dolnoslaskie():
10     population = 2902365
11     print('Liczba mieszkańców województwa dolnośląskiego to ',\
12           population,'.')
13
14 #Funkcja mazowieckie tworzy lokalną zmienną population.
15 def mazowieckie():
16     population = 5391813
17     print('Liczba mieszkańców województwa mazowieckiego to ',\
18           population,'.')
19
20 main()
21
```

Politechnika Wrocławska

Funkcje

Przekazywanie argumentów do funkcji

pass_argument.py

```python
#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()
```

Politechnika Wrocławska

Funkcje

Przekazywanie argumentów do funkcji

pass_argument.py

```python
#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()
```

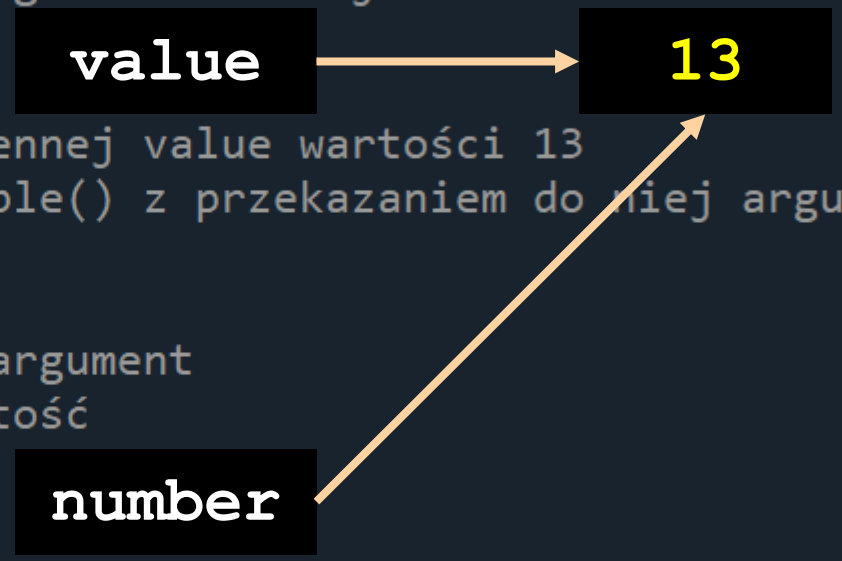
Politechnika Wrocławska

Funkcje

Przekazywanie argumentów do funkcji

pass_argument.py

```
1   #Program pokazuje przekazanie argumentu funkcji.
2
3   def main():
4       value = 13 #przypisanie zmiennej value wartości 13
5       #wywołanie funkcji show_double() z przekazaniem do niej argumentu
6       show_double(value)
7
8   #Funkcja show_double() pobiera argument
9   #i wyświetla jego podwojoną wartość
10  def show_double(number):
11      result = number * 2
12      print(result)
13
14  main()
15
```



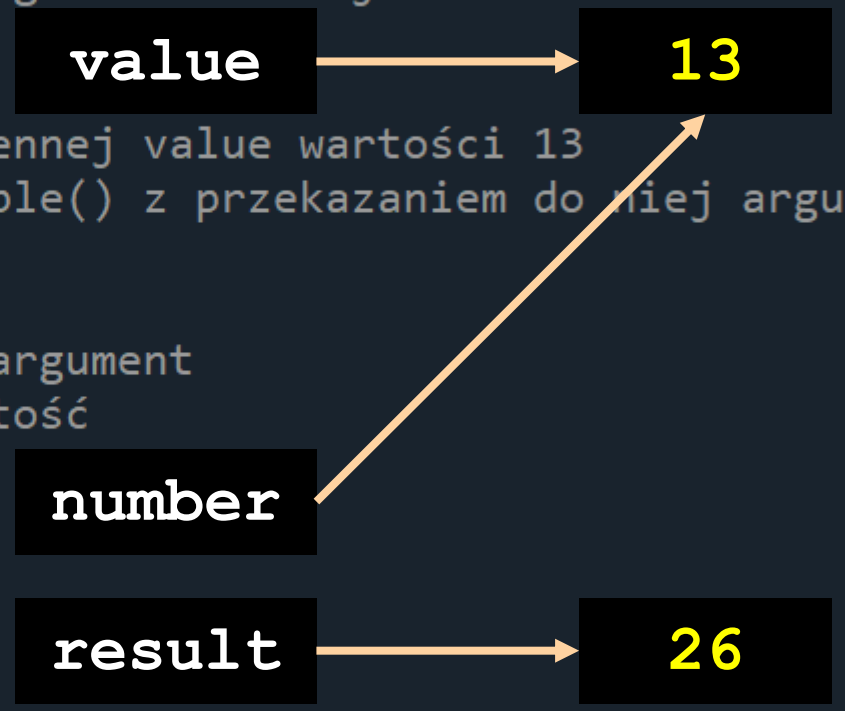
Politechnika Wrocławska

Funkcje

Przekazywanie argumentów do funkcji

pass_argument.py

```
1   #Program pokazuje przekazanie argumentu funkcji.
2
3   def main():
4       value = 13 #przypisanie zmiennej value wartości 13
5       #wywołanie funkcji show_double() z przekazaniem do niej argumentu
6       show_double(value)
7
8   #Funkcja show_double() pobiera argument
9   #i wyświetla jego podwojoną wartość
10  def show_double(number):
11      result = number * 2
12      print(result)
13
14  main()
15
```



Politechnika Wrocławska

Funkcje

Przekazywanie argumentów do funkcji

multiple_arguments.py

```
1   #Program pokazuje przekazanie dwóch argumentów do funkcji.
2
3   def main():
4       print('Suma liczb 12 i 45 wynosi')
5       #Wartości wyrażeń użytych w miejscu wywołania
6       #zostaną nadane zmiennym w funkcji show_sum().
7       show_sum(12,45)
8
9   #Funkcja show_sum() pobiera dwa argumenty
10  #i wyświetla ich sumę.
11  #W nawiasie umieszczono listę argumentów.
12  #Nazwy argumentów są rozdzielone przecinkiem.
13  def show_sum(num1, num2):
14      result = num1 + num2
15      print(result)
16
17  main()
```

Politechnika Wrocławska

Funkcje

Zmiana argumentu funkcji

change_me.py

```
1  #Program pokazuje, co się stanie po zmianie
2  #wartości argumentu w funkcji.
3
4  def main():
5      value = 99
6      print('Wartość wynosi',value)
7      change_me(value)
8      print('Wartość w funkcji main() wynosi',value)
9
10 def change_me(arg):
11     print('Do funkcji change_me() przekazano wartość',arg)
12     arg = 0
13     print('Przypisano nową wartość do zmiennej arg')
14     print('Teraz wartość wynosi',arg)
15
16 main()
```

Console 1/A

```
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
```

Politechnika Wrocławska

Funkcje

Przekazywanie argumentów do funkcji

- Przedstawiony sposób przekazywania argumentów do funkcji jest jednym z kilku możliwych (przez pozycje, przez nazwę, argumenty domyślne).

Politechnika Wrocławska

Funkcje

Funkcje zwracające wartość

- Prosta postać funkcji zwracającej wartość

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

Politechnika Wrocławska

Funkcje

Funkcje zwracające wartość

return_result_01.py

```
1  #Program pokazuje definicję i wywołanie funkcji zwracajacej wartość.
2
3  #Program pyta użytkownika o liczbę jabłek i pomarańczy.
4  #Następnie wywołuje funkcję sum() do obliczenia łącznej liczby owoców
5  #i wyświetla stosowny komunikat.
6  def main():
7      apples  = int(input('Ile masz jabłek? '))
8      oranges = int(input('Ile masz pomarańczy? '))
9      print('Liczba wszystkich owoców',sum(apples,oranges))
10
11 #Funkcja sum oblicza sumę dwóch liczb
12 def sum(num1, num2):
13     result = num1 + num2 #zapisanie wyniku w zmiennej pomocniczej
14     return result        #zwrócenie wyniku
15
16 main()
```

Console 1/A

```
Ile masz jabłek? 5
Ile masz pomarańczy? 6
Liczba wszystkich owoców 11
```

Politechnika Wrocławska

Funkcje

Funkcje zwracające wartość

return_result_01.py | return_result_02.py

```
1  #Program pokazuje definicję i wywołanie funkcji zwracajacej wartość.
2
3  #Program pyta użytkownika o liczbę jabłek i pomarańczy.
4  #Następnie wywołuje funkcję sum() do obliczenia łącznej liczby owoców
5  #i wyświetla stosowny komunikat.
6  def main():
7      apples  = int(input('Ile masz jabłek? '))
8      oranges = int(input('Ile masz pomarańczy? '))
9      print('Liczba wszystkich owoców',sum(apples,oranges))
10
11 #Funkcja sum oblicza sumę dwóch liczb
12 def sum(num1, num2):
13     return num1 + num2 #polecenie return może zwracać wartość wyrażenia
14
15 main()
16
```

Console 1/A

```
Ile masz jabłek? 5
Ile masz pomarańczy? 6
Liczba wszystkich owoców 11
```

Politechnika Wrocławska

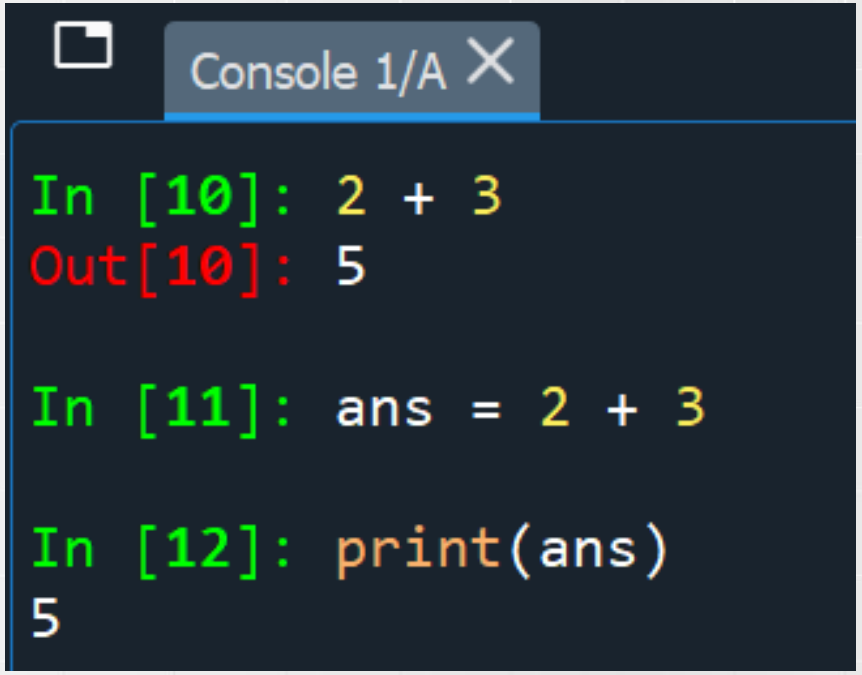
Operatory matematyczne

symbol	działanie
+	dodawanie
-	odejmowanie
*	mnożenie
/	dzielenie
//	dzielenie całkowite
%	reszta z dzielenia
**	potęgowanie

Politechnika Wrocławska

Operatory matematyczne

- Operator matematyczny wykonuje działanie i zwraca wynik
- Wynik można przypisać do zmiennej



Politechnika Wrocławska

Operatory matematyczne

Kolejność wykonywania działań

1. potęgowanie
2. mnożenie, dzielenie, dzielenie całkowite, reszta z dzielenia
3. dodawanie, odejmowanie

Politechnika Wrocławska

Operatory matematyczne

Kolejność wykonywania działań

Po wykonaniu polecenia

```
ans = 5 + 2 * 3
```

wartość zmiennej **`ans`** to 11

Politechnika Wrocławska

Operatory matematyczne

Kolejność wykonywania działań

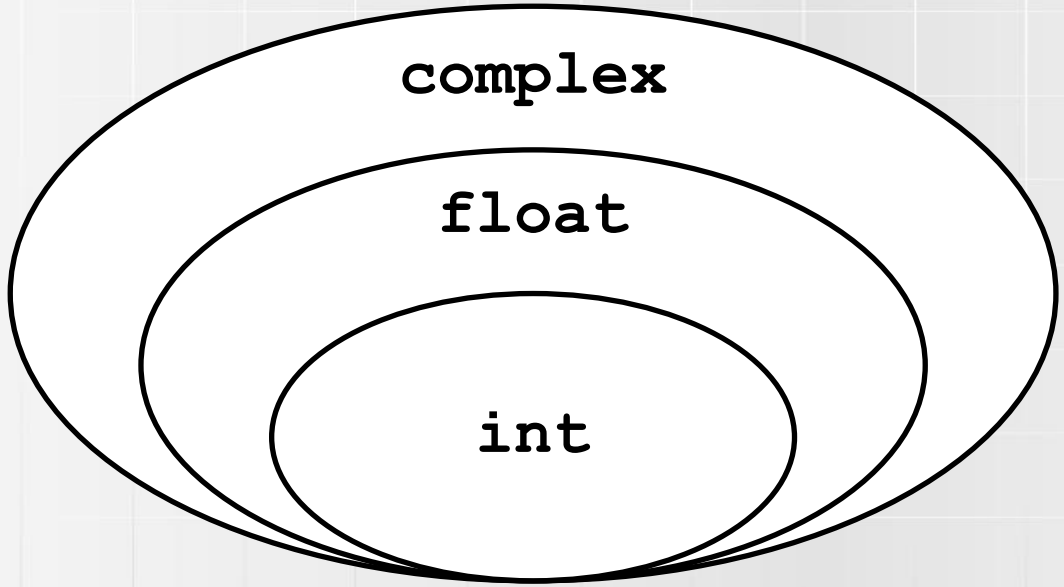
```
ans = 5 + 2 * 4 / 2 % 3 + 10 - 3

5 + 8 / 2 % 3 + 10 - 3
    5 + 4 % 3 + 10 - 3
        5 + 1 + 10 - 3
            6 + 10 - 3
                16 - 3
                   13
```

Politechnika Wrocławska

Operatory matematyczne

Różne typy danych

- Typ danych wyniku zależy od typów danych argumentów
- Konwersja następuje do „szerszego” typu danych

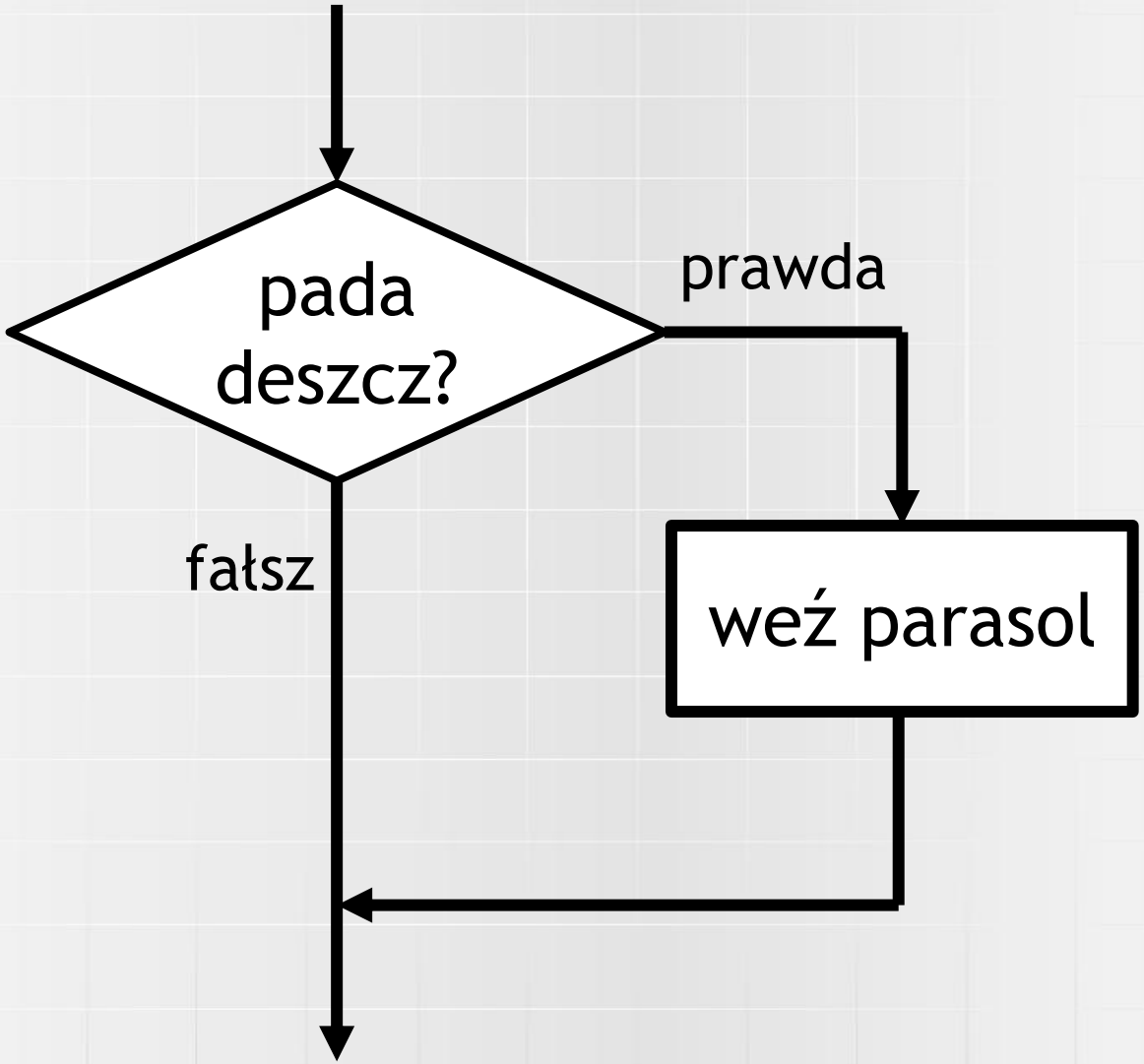


Instrukcja warunkowa

- Instrukcja warunkowa pozwala na uzyskanie więcej niż jednej ścieżki wykonywania programu

Politechnika Wrocławska

Instrukcja warunkowa



Politechnika Wrocławska

Operatory relacji

- Operator relacji służy do sprawdzenia, czy między dwiema wartościami zachodzi określona relacja

Politechnika Wrocławska

Operatory relacji

Operator	Opis
<	mniejsze niż
>	większe niż
<=	mniejsze lub równe
>=	większe lub równe
==	równe
!=	nierówne

Politechnika Wrocławska

Instrukcja warunkowa `if`

Przykład 1

example_if_01.py

```
1   #Program pokazuje użycie instrukcji warunkowej.
2   #Program pobiera od użytkownika punktację (liczbę całkowitą)
3   #i wyświetla gratulacje, jeśli wynik jest większy
4   #lub równy 50 pkt.
5
6   def main():
7       rate = int(input('Podaj swoją punktację [0-100]: '))
8       if rate >= 50:
9           print('Gratulacje!')
10
11  main()
```

Console 1/A

```
In [19]: runfile('C:/Dane Karola/repozytoria/python-examples/if/examp
Podaj swoją punktację [0-100]: 77
Gratulacje!

In [20]: runfile('C:/Dane Karola/repozytoria/python-examples/if/examp
Podaj swoją punktację [0-100]: 48
```

Politechnika Wrocławska

Instrukcja warunkowa `if`

Przykład 2

example_if_02.py

```
1   #Program pokazuje użycie instrukcji warunkowej.
2   #W bloku instrukcji if znajduje się kilka poleceń.
3
4   #Program pobiera od użytkownika punktację (liczbę całkowitą)
5   #i wyświetla gratulacje, jeśli wynik jest większy
6   #lub równy 50 pkt.
7
8   def main():
9       rate = int(input('Podaj swoją punktację [0-100]: '))
10      if rate >= 50:
11          print('Gratulacje!')
12          print('Świetny rezultat!')
13
14  main()
```

Console 1/A

```
In [21]: runfile('C:/Dane Kurotu/repozytoria/python-examples/tj/examp
Podaj swoją punktację [0-100]: 95
Gratulacje!
Świetny rezultat!
```

Politechnika Wrocławska

Instrukcja warunkowa `if`

Przykład 3

example_if_03.py

```
1   #Program pokazuje użycie instrukcji warunkowej.
2   #Zakończenie bloku poleceń zależnych od warunku
3   #osiąga się odpowiednimi wcięciami.
4
5   #Program pobiera od użytkownika punktację (liczbę całkowitą)
6   #i wyświetla gratulacje, jeśli wynik jest większy
7   #lub równy 50 pkt.
8
9   def main():
10      rate = int(input('Podaj swoją punktację [0-100]: '))
11
12      if rate >= 50:
13          print('Gratulacje!')
14          print('Świetny rezultat!')
15
16      print('Niezależnie od wyniku, dzięki za udział.')
17
18  main()
19
```

Politechnika Wrocławska

Instrukcja warunkowa `if`

Przykład 3

example_if_03.py

```
1   #Program pokazuje użycie instrukcji warunkowej.
2   #Zakończenie bloku poleceń zależnych od warunku
3   #osiąga się odpowie
4
5   #Program pobiera od
6   #i wyświetla gratul
7   #lub równy 50 pkt.
8
9   def main():
10      rate = int(input('Podaj swoją punktację [0-100]: '))
11
12      if rate >= 50:
13          print('Gratulacje!')
14          print('Świetny rezultat!')
15
16      print('Niezależnie od wyniku, dzięki za udział.')
17
18  main()
19
```

Console 1/A

```
Podaj swoją punktację [0-100]: 65
Gratulacje!
Świetny rezultat!
Niezależnie od wyniku, dzięki za udział.
```

Politechnika Wrocławska

Instrukcja warunkowa `if`

Przykład 3

example_if_03.py

```
1   #Program pokazuje użycie instrukcji warunkowej.
2   #Zakończenie bloku poleceń zależnych od warunku
3   #osiąga się odpowie
4
5   #Program pobiera od
6   #i wyświetla gratul
7   #lub równy 50 pkt.
8
9   def main():
10      rate = int(inpu
11
12      if rate >= 50:
13          print('Grat
14          print('Świe
15
16      print('Niezależnie od wyniku, dzięki za udział.')
17
18  main()
19
```

Console 1/A

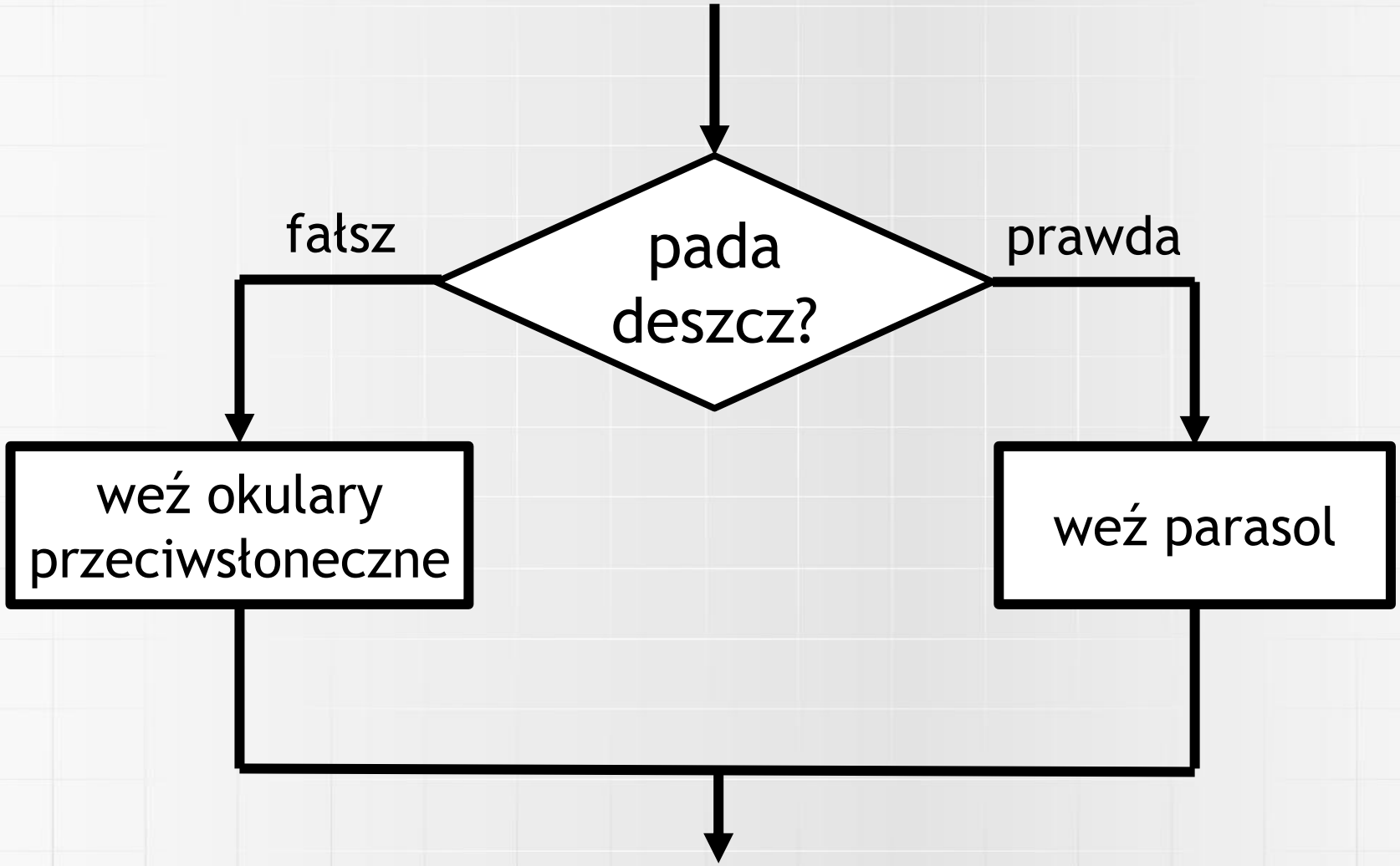
```
Podaj swoją punktację [0-100]: 65
Gratulacje!
Świetny rezultat!
Niezależnie od wyniku, dzięki za udział.
```

Console 1/A

```
Podaj swoją punktację [0-100]: 32
Niezależnie od wyniku, dzięki za udział.
```

Politechnika Wrocławska

Instrukcja warunkowa `if-else`



Politechnika Wrocławska

Instrukcja warunkowa `if-else`

- Instrukcja **`if-else`** nazywana jest instrukcją warunkową podwójnego wyboru

Politechnika Wrocławska

Instrukcja warunkowa `if-else`

Przykład

example_if_04.py

```
#Program pokazuje użycie instrukcji warunkowej
#podwójnego wyboru.

#Program pobiera od użytkownika punktację (liczbę całkowitą)
#
#Jeśli wynik jest większy lub równy 50 pkt.
#   wyświetla gratulacje,
#w przeciwnym przypadku
#   informuje złym wyniku.
#
#Następnie dziękuje za udział.
#
def main():
    rate = int(input('Podaj swoją punktację [0-100]: '))

    if rate >= 50:
        print('Gratulacje!')
        print('Świetny rezultat!')
    else:
        print('Twój wynik nie jest dobry!')
        print('Powodzenia następnym razem')

    print('Niezależnie od wyniku, dzięki za udział.')

main()
```

Politechnika Wrocławska

Instrukcja warunkowa `if-else`

Przykład

example_if_04.py

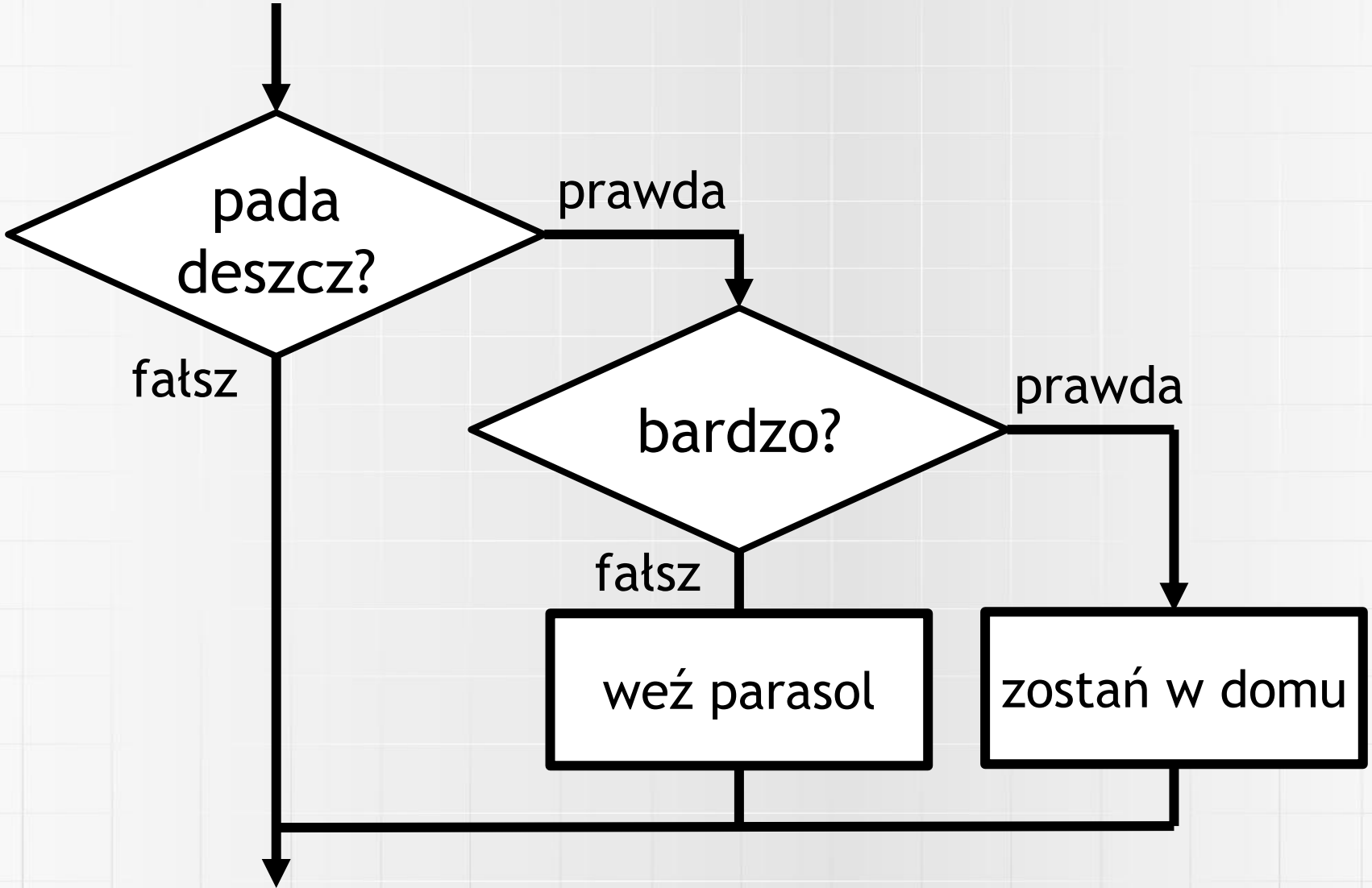
```
1  #Program pokazuje użycie instrukcji warunkowej
2  #podwójnego wyboru.
3
4  #Program pobiera od użytkown
5  #
6  #Jeśli wynik jest większy lu
7  #   wyświetla gratulacje,
8  #w przeciwnym przypadku
9  #   informuje złym wyniku.
10 #
11 #Następnie dziękuje za udzia
12 #
13 def main():
14     rate = int(input('Podaj
15
16     if rate >= 50:
17         print('Gratulacje!')
18         print('Świetny rezul
19     else:
20         print('Twój wynik ni
21         print('Powodzenia następnym razem')
22
23     print('Niezależnie od wyniku, dzięki za udział.')
24
25 main()
```

Console 1/A

```
Podaj swoją punktację [0-100]: 55
Gratulacje!
Świetny rezultat!
Niezależnie od wyniku, dzięki za udział.

In [25]: runfile('C:/Dane Karola/repozytoria
Podaj swoją punktację [0-100]: 45
Twój wynik nie jest dobry!
Powodzenia następnym razem
Niezależnie od wyniku, dzięki za udział.
```

Politechnika Wrocławska

Zagnieżdżone instrukcje warunkowe



Politechnika Wrocławska

Zagnieżdżone instrukcje warunkowe

Przykład

example_if_05.py

```
1   #Program pokazuje użycie zagnieżdżonych instrukcji warunkowych
2
3   #Program pobiera od użytkownika punktację (liczbę całkowitą)
4   #
5   #Jeśli wynik jest większy lub równy 50 pkt.
6   #   wyświetla gratulacje.
7   #   Dodatkowo, jeśli wynik jest większy lub równy 90 pkt.
8   #       wyświetla informację o świetnym wyniku.
9   #   w przeciwnym wypadku
10  #       wyświetla motywujący komunikat.
11  #w przeciwnym przypadku
12  #   informuje złym wyniku.
13  #   Dodatkowo, jeśli wynik jest większy lub równy 40 pkt.
14  #       wyświelta informację, że zabrakło niewiele.
15  #   w przeciwnym przypadku
16  #       wyświetla motywujący komunikat.
17  #
```

Politechnika Wrocławska

Zagnieżdżone instrukcje warunkowe

Przykład

example_if_05.py

```
18  def main():
19      rate = int(input('Podaj swoją punktację [0-100]: '))
20
21      if rate >= 50:
22          print('Gratulacje!')
23          if rate >= 90:
24              print('Świetny wynik!')
25          else:
26              print('Możesz się jeszcze poprawić!')
27      else:
28          print('Twój wynik nie jest dobry!')
29          if rate >= 40:
30              print('Zabrakło niewiele')
31          else:
32              print('Dużo pracy przed Tobą!')
33
34  main()
35
```

Politechnika Wrocławska

Instrukcja warunkowa

`if-elif-else`

example_if_06.py

```
#Program pokazuje użycie instrukcji warunkowej if-elif-else.

#Program pobiera od użytkownika punktację (liczbę całkowitą)
#
#Jeśli wynik jest większy lub równy 90 pkt.
#       wyświetla informację o świetnym wyniku,
#jeśli wynik jest mniejszy niż 90 pkt., ale większy równy 50
#       wyświetla motywujący komunikat,
#jeśli wynik jest mniejszy niż 50, ale większy nić 40 pkt.
#       wyświelta informację, że zabrakło niewiele,
#w przeciwnym przypadku
#       wyświetla motywujący komunikat.
#
def main():
    rate = int(input('Podaj swoją punktację [0-100]: '))

    if rate >= 90:
        print('Świetny wynik!')
    elif rate >= 50:
        print('Możesz się jeszcze poprawić!')
    elif rate >= 40:
        print('Zabrakło niewiele')
    else:
        print('Dużo pracy przed Tobą!')

main()
```

Politechnika Wrocławska

Operatory logiczne

Operator	Opis
`and`	koniunkcja
`or`	alternatywa
`not`	negacja

Politechnika Wrocławska

Operatory logiczne

example_and_or_not.py

```
1   #Program pokazuje działanie operatorów logicznych.
2
3   def main():
4       x = int(input("Podaj liczbę całkowitą: "))
5
6       if x > 20 and x < 40:
7           print("Liczba mieści się w przedziale (20,40).")
8
9       if x <= 20 or x >= 40:
10          print("Liczba nie mieści się w przedziale (20,40).")
11
12      if x != 30:
13          print("Liczba jest różna od 30.")
14
15      if not x == 30:
16          print("Liczba jest różna od 30.")
17
18
19  main()
```

Politechnika Wrocławska

Podsumowanie

Algorytmy

- Algorytmy i programy
- Algorytmy liniowe
- Algorytmy z rozgałęzieniami

Podstawowe funkcje i elementy języka python

- Funkcja print()
- Ciągi tekstowe
- Komentarze
- Zmienne
- Literały liczbowe
- Funkcja input()
- Funkcje
- Operatory matematyczne
- Instrukcja warunkowa i operatory logiczne