Wrocław University of Science and Technology

Wstęp do programowania

INP003203L

Semestr zimowy 2020/2021

Poniedziałek, 7:30 - 9:00

sala wirtualna
– zajęcia online

Sylwia Majchrowska

sylwia.majchrowska@pwr.edu.pl

https://majsylw.netlify.app/teaching/

pokój 213, budynek L-1

Wrocław University of Science and Technology

Plan prezentacji

1. Operacje matematyczne:
   - operatory arytmetyczne,
   - operatory binarne i unarne,
   - operowanie na napisach,
   - kolejność wykonywania działań.
2. Funkcje:
   - definiowanie funkcji (*i procedur*),
   - przekazywanie argumentów,
   - zwracanie wyniku.
3. Operacje wyjścia – funkcja **print**():
   - formatowanie danych wyjściowych.

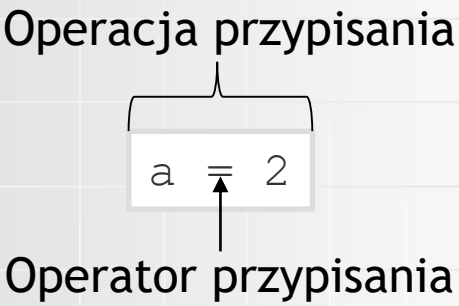
Wrocław University of Science and Technology

Operatory arytmetyczne

Operator to symbol wykorzystywany w danym języku programowania, aby w konkretny sposób zadziałać na występujące przy nim wartości lub zmienne.

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

```
>> print(2+2)
```



Działanie	Operator
Dodawanie	+
Odejmowanie	-
Mnożenie	*
Dzielenie	/
Dzielenie całkowitoliczbowe	//
Modulo	%
Potęgowanie	**

Wrocław University of Science and Technology

Operatory arytmetyczne

- potęgowanie

Operator ** odpowiada za potęgowanie. Przed nim umieszczamy podstawę potęgowania, a za wykładnik.

~~2^3, 2^3~~

```
print(2 ** 3)
print(2 ** 3.)
print(2. ** 3)
print(2. ** 3.)
```

Zapamiętaj!

- Kiedy oba argumenty dla operacji ** są całkowite, jej wynik także będzie całkowity
- Gdy przynajmniej jeden z argumentów jest zmiennoprzecinkowy, wynik będzie zmiennoprzecinkowy (float)

Wrocław University of Science and Technology

Operatory arytmetyczne

- mnożenie, dzielenie i jego reszta

- Operator * odpowiada za pomnożenie dwóch liczb - iloczyn.
- Operator / odpowiada za podzielenie dwóch liczb - iloraz.
- Operator // odpowiada za dzielenie całkowite dwóch liczb - iloraz.
- Operator % odpowiada za zwrócenie reszty z dzielenia - modulo

```
print(2 * 3)
print(2 * 3.)
print(2. * 3)
print(2. * 3.)
```

```
print(6 / 3)
print(2 / 1)
print(6 / 3.)
print(6. / 3)
print(6. / 3.)
```

```
print(6 // 3)
print(6 // 3.)
print(6. // 3)
print(6. // 3.)
print(6 // 4)
print(6. // 4)
print(-6 // 4)
```

```
print(14 % 4)
print(14. % 4)
print(12 % 4.5)
```

Dzielenie przez 0?

Zapamiętaj!

- Typ wyniku dla mnożenia zachowa się tak samo jak dla potęgowania.
- Rezultat dzielenia / jest zawsze zmiennoprzecinkowy (float).
- Typ wyniku dzielenia całkowitoliczbowego // zależy od typów jego argumentów (jak powyżej mnożenie).
- Przy dzieleniu całkowitoliczbowym // wyniki są zaokrąglone w dół.
- Typ wyniku operacji modulo % zależy od typów jej argumentów (jak powyżej mnożenie).

Wrocław University of Science and Technology

Operatory arytmetyczne

- dodawanie, odejmowanie i operatory unarne

- Operator + jako operator binarny odpowiada za dodawanie 2 liczb, jako operator unarny określa liczbę dodatnią.
- Operator - jako operator binarny odpowiada za dodawanie 2 liczb, jako operator unarny określa liczbę ujemną.

```
print(-4 + 4)
print(-4. + 8)
print(-4 - 4)
print(4. - 8)
print(-1.1)
print(+2)
```

Zapamiętaj!

- Kiedy oba argumenty dla operacji + lub - są całkowite, jej wynik także będzie całkowity
- Gdy przynajmniej jeden z argumentów jest zmiennoprzecinkowy, wynik będzie zmiennoprzecinkowy (float)

Wrocław University of Science and Technology

Operatory arytmetyczne

- operacje na napisach: konkatenacja i powtarzanie

- Operator + gdy dwa podane mu argumenty są napisami złączy je ze sobą (konkatenacja).
- Operator * gdy jeden z podanych mu argumentów jest napisem powtórzy napis odpowiednią liczbę razy (replikacja).
- Konkatenacja nie jest przemienna, replikacja jest przemienna.

```
print("2" * 5) # "22222"
```

Ćwiczenie 3.1 – narysuj prostokąt:

```
+----------+
|          |
|          |
|          |
|          |
|          |
+----------+
```

```
print("+" + 10 * "-" + "+")
print(("|" + " " * 10 + "|\n") * 5, end="")
print("+" + 10 * "-" + "+")
```

Wrocław University of Science and Technology

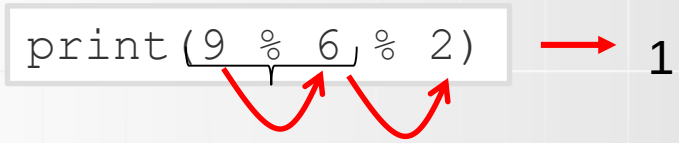
Operatory arytmetyczne

- kolejność wykonywania działań

1. Wyrażenia w nawiasach
2. Potęgowanie **
3. Mnożenie *, dzielenie /, dzielenie całkowite //, reszta z dzielenia %
4. Dodawanie +, odejmowanie -

Co z wyrażeniami równorzędnymi?

```
print(9 % 6 % 2)
```

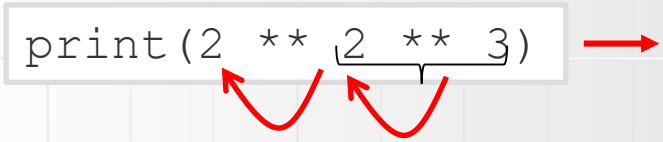


→ 1

Większość operatorów arytmetycznych działa od lewej do prawej.

Ale...

```
print(2 ** 2 ** 3)
```



→ 256

Operator ** działa od prawej do lewej.

Wrocław University of Science and Technology

Operatory arytmetyczne

- kolejność wykonywania działań – rozwiązania 3.2

Zad 1. Podaj wynik końcowy wyrażenia: 5 + 2 * 4 / 2 % 3 + 10 - 3

```
print(5 + 2 * 4 / 2 % 3 + 10 - 3) # 13.0
```

Zad 2. Przyjmij a = 23 oraz b = 5. Wylicz sumę, iloczyn, iloraz oraz resztę z dzielenia dla tych zmiennych.

```
a = 23
b = 5
print(a + b) # 28
print(a - b) # 18
print(a * b) # 115
print(a / b) # 4.6
print(a // b) # 4
print(a % b) # 3
```

```
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
```

Zad 3. Wylicz średnią dla liczb: 3, 6, 3.

```
print((3 + 6 + 3) / 3) # 4.0
```

Zad 4. Jaki będzie wynik działania: 2 * 3 % 5?

```
print(2 * 3 % 5) # 1
```

Wrocław University of Science and Technology

Operatory arytmetyczne

- kolejność wykonywania działań – rozwiązanie 3.3

Napisz program który pozwoli na wyznaczenie wartości dla wyrażenia

$$\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x}}}}$$

```
x = float(input("Wprowadź wartość x: "))
y = 1/(
    x + 1/(
        x + 1/(
            x + 1/(x))))
print("y = ", y)
```

Wrocław University of Science and Technology

Funkcje

- samodzielne definiowanie funkcji

definicja funkcji suma()

```
def suma(a, b):
    s = a + b
    return s
```

dwukropek

nagłówek funkcji suma()

ciało funkcji suma()
(wydzielone przez 4 spacje lub 1 tabulator)

nazwa funkcji

argumenty

słowa kluczowe

zwracane wartości

wywołanie funkcji suma()

```
suma(2,3)
s1 = 4
s2 = 5
suma(s1, s2)
```

```
>>> type(suma)
<class 'function'>
```

```
>>> type(print)
<class 'builtin_function_or_method'>
```

Funkcja to oddzielna część kodu komputerowego, która może:

- **wywołać jakiś efekt** (np. wysłać tekst do terminala, stworzyć plik, narysować obrazek, odtworzyć dźwięk itp.); jest to coś zupełnie niespotykanego w świecie matematyki;
- **obliczyć jakaś wartość** lub wartości (np. pierwiastek kwadratowy z wartości lub długość danego tekstu); to właśnie sprawia, że funkcje programistyczne są krewnymi pojęć matematycznych.

Wrocław University of Science and Technology

Funkcje

- jak działają funkcje?

```
def wiadomosc():
    print("Jesteś wewnątrz funkcji.")

print(„Jeszcze nie wywołano funkcji.")
wiadomosc()
print("Wyszedłem z funkcji!")
```

return

wywołanie

Nie możemy wywoływać funkcji, które nie zostały zdefiniowane!

```
print("Zaczynamy.")
wiadomosc()
print("Kończymy.")

def wiadomosc():
    print("Jesteś w funkcji")
```

NameError: name ‚wiadomosc' is not defined

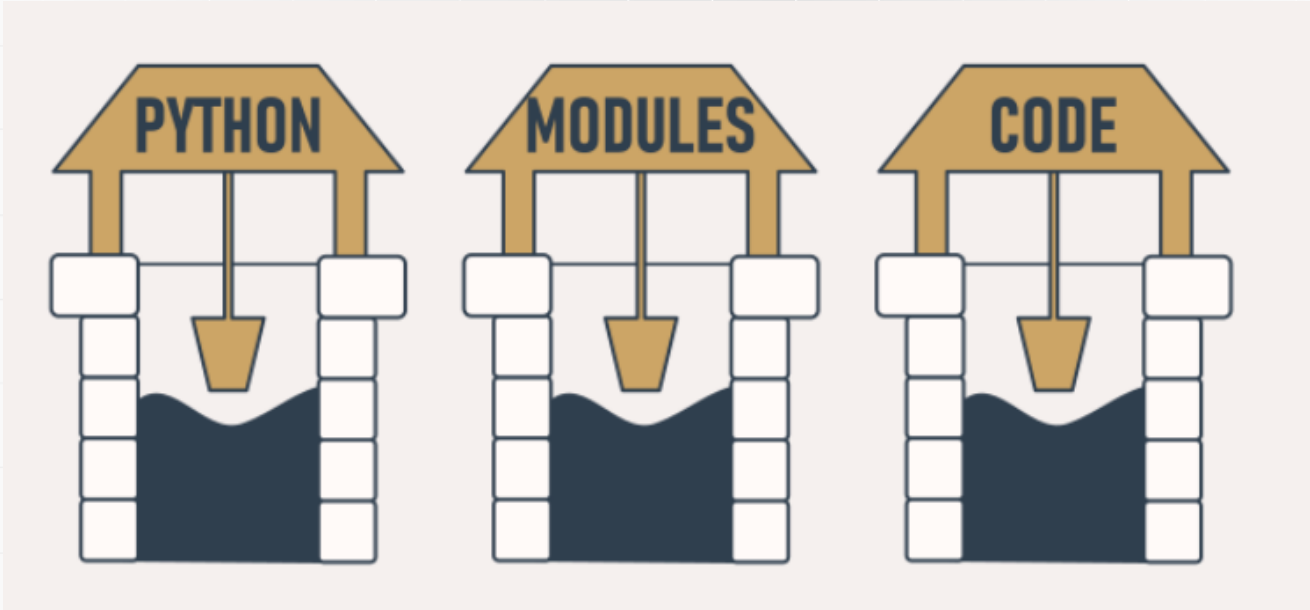
Wrocław University of Science and Technology

Funkcje

- po co nam 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



Wrocław University of Science and Technology

Funkcje

- przekazywanie argumentów

argumenty domyślne

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

```
print("a", "b", sep="***", end=" ")
```

- Przekazywanie argumentów względem ich pozycji (**argumenty pozycyjne**):
  - Kolejność przekazywanych argumentów musi opowiadać ich kolejności w definicji funkcji; wszelkie argumenty nazwane należy umieścić po ostatnim argumencie pozycyjnym.
- Przekazywanie argumentów względem ich nazwy (**argumenty nazwane**):
  - Przekazywany argument nazwany składa się z 3 części – nazwy argumentu, operatora przypisania (=) oraz wartości przypisywanej do tego argumentu

UWAGA!!!

- **argumenty** są znane jedynie wewnątrz funkcji, w której zostały wywołane,
- zmienne zdefiniowane wewnątrz funkcji (**zmienne lokalne**) ‚żyją' tylko w jej obrębie, ale mogą zostać **przykryte** (ang. shadowing),
- **stałe nazwane** mogą być wykorzystane w każdym punkcie programu.

global

Wrocław University of Science and Technology

Funkcje

- przekazywanie argumentów – rozwiązania 3.4

```
def dodawanie(a, b, c):
    print(a, "+", b, "+", c, "=", a + b + c)
```

Co się stanie po wywołaniu funkcji dodawanie?

a) dodawanie(1, 2, 3)

```
"1 + 2 + 3 = 6"
```

b) dodawanie(c = 1, a = 2, b = 3)

```
"2 + 3 + 1 = 6"
```

c) dodawanie(3, c = 1, b = 2)

```
"3 + 2 + 1 = 6"
```

d) dodawanie(3, a = 1, b = 2)

```
TypeError: dodawanie() got
multiple values for argument 'a'
```

e) dodawanie(1, 3, c = 2)

```
"1 + 3 + 2 = 6"
```

f) dodawanie(c = 4, 3, 2)

```
SyntaxError: positional argument
follows keyword argument
```

g) spróbuj teraz zmodyfikować definicję funkcji tak aby jeden z argumentów miał wartość domyślną.

```
def dodawanie(a, b, c = 4):
    …
```

Wrocław University of Science and Technology

Funkcje

- zwracanie wartości i ich przypisywanie do zmiennych

Funkcja to oddzielna część kodu komputerowego, która może:

- **zwracać konkretne wartości** (instrukcja return <sth>)
- **wykonywać konkretne operacje** (pusta instrukcja return lub bez tego słowa kluczowego) – zwracać ***None*** (brak wartości)

UWAGA!!!

- zawsze możesz **zignorować** rezultat funkcji,
- jeśli funkcja ma **zwrócić jakąś wartość** powinna zawierać słowo kluczowe **return**.
- jeśli funkcja **ma zwrócić kilka wartości** należy rozdzielić je przecinkiem (i również odpowiednio przypisać ich wartości do zmiennych)

Wrocław University of Science and Technology

Funkcje

- projektowanie programu: funkcja main()

```
print("Zaczynamy.")
wiadomosc()
print("Kończymy.")

def wiadomosc():
    print("Jesteś w funkcji")
```

NameError: name 'wiadomosc' is not defined

Funkcje mogą być definiowane w dowolnej kolejności.

```
def main():
  print("Zaczynamy.")
  wiadomosc()
  print("Kończymy.")
def wiadomosc():
  print("Jesteś w funkcji")
main()
```

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

Wrocław University of Science and Technology

Funkcja print()

- formatowanie napisów

Znaki sterujące:

- Znak specjalny końca linii *'\n'*,
- Znak specjalny tabulacji *'\t'*,
- Znak specjalny backspace *'\b'*,
- Znak apostrofu \',
- Znak cudzysłowu \",
- Ukośnik \\.

```
format(value[, format_spec])
```

Funkcja format():

- pozwala na wyświetlenie znaku przy liczbie
- pozwala na ustalenie liczby miejsc po przecinku dla liczb zmiennoprzecinkowych
- pozwala na zachowanie odpowiedniego odstępu dla liczb oraz napisów

Typ	Znaczenie
'E', 'e'	Notacja naukowa
'F', 'f'	Liczby zmiennoprzecinkowe
'G', 'g', None	Automatycznie ustala czy wyświetlić coś jak 'f' czy ‚e' (tryb domyślny)
'D', 'd'	Liczby całkowite
'%'	Wartość procentowa
'S', 's'	Napis

https://docs.python.org/3/library/string.html#formatspec

Wrocław University of Science and Technology

Funkcja print()

– formatowanie wyniku – rozwiązanie 3.5

Napisz program wypisujący na ekranie tabelę przedrostków SI zawierającą w kolejnych kolumnach symbol, nazwę, wykładnik mnożnika oraz mnożnika od femto- do peta-. Efekt działania programu powinien być następujący:

symbol	nazwa	wykladnik	mnoznik
f	femto	-15	1e-015
p	piko	-12	1e-012
n	nano	-9	1e-009
mu	mikro	-6	1e-006
m	mili	-3	0.001
c	centy	-2	0.01
d	decy	-1	0.1
da	deka	1	10
h	hekto	2	100
k	kilo	3	1000
M	mega	6	1e+006
G	giga	9	1e+009
T	tera	12	1e+012
P	peta	15	1e+015

```
print(format('symbol','6s'),
      format('nazwa','6s'),
      format('wykładnik','9s'),
      format('mnożnik','7s'))
print(format('f','6s'),
      format('femto','6s'),
      format(-15,'<9d'),
      format(1e-15,'7.0g'))
…
print(format('f','6s'),
      format('kilo','6s'),
      format(3,'<9d'),
      format(1e3,'7.0f'))
```

https://stackoverflow.com/questions/9910972/number-of-digits-in-exponent

Wrocław University of Science and Technology

Formatowanie wyniku

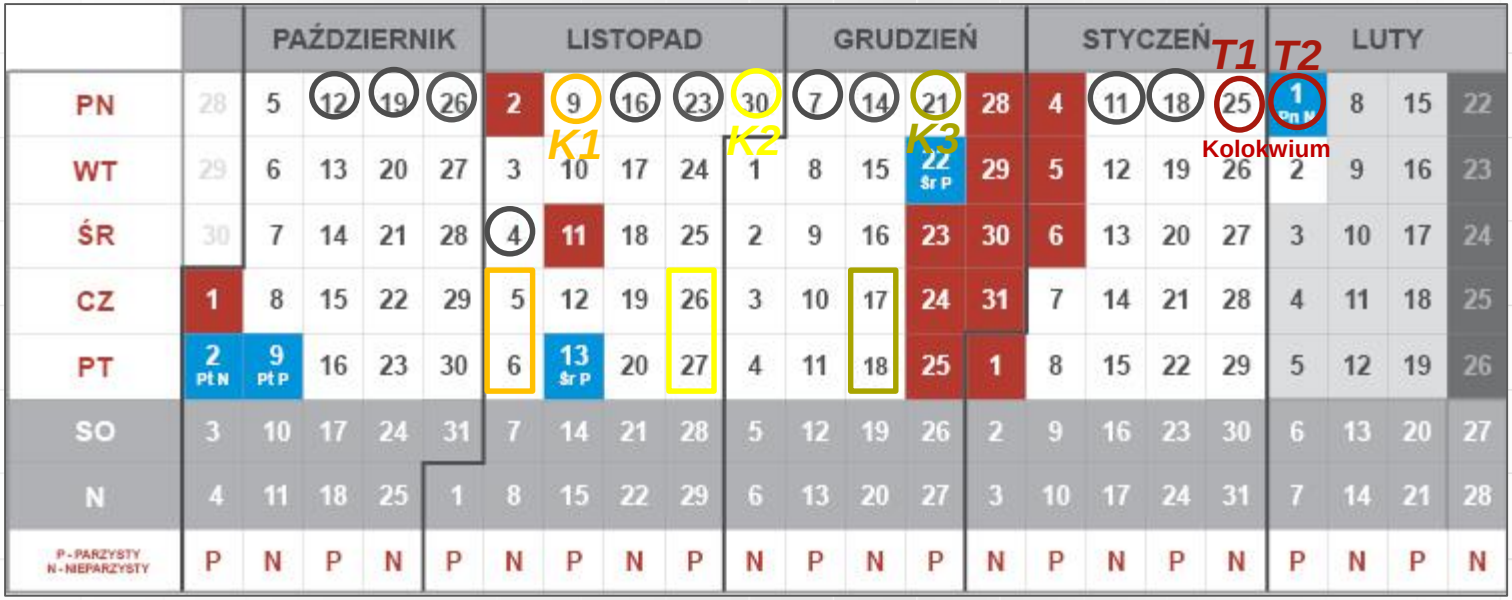
- rozwiązanie przykładu 3.6

Napisz program który pozwoli na wyznaczenie godziny (w formacie hh:gg) końca spotkania na podstawie podanej wartości godziny (i minut) jego początku oraz czasu trwania w minutach.

```
def main():
  hour = int(input("Czas startu (godziny): "))
  mins = int(input("Czas startu (minuty): "))
  dura = int(input("Czas trwania (minuty): "))

  hour = hour + dura // 60
  mins = mins + dura % 60

  hour = (hour + mins // 60) % 24
  mins = mins % 60

  print(format(hour, "02d"), ":", format(mins, "02d"))
```

Wrocław University of Science and Technology

Spojrzenie na kalendarz



Poniedziałek, 7:30 - 9:00
sala wirtualna

Konsultacje:
Poniedziałki, 19:30 - 20:30
Środy, 19:30 – 20:30

Wrocław University of Science and Technology

Rzeczy do zapamiętania!

1. Operatory to specjalne symbole, które pozwalają na wykonanie szeregu operacji, np. operator mnożenia *, przemnaża dwie wartości.
2. Wyrażenia to kombinacja zmiennych, wartości i operatorów, np. 1 + 2.
3. Kolejność wykonywania działań w pythonie jest taka sama jak znana nam z matematyki: unarny + i -, potem potęgowanie – mnożenie, dzielenie, modulo – dodawanie i odejmowanie.
4. Operatory arytmetyczne działają od lewej do prawej z wyjątkiem operatora potęgowania - działa od prawej do lewej.
5. Operacje w nawiasach zawsze są wykonywane jako pierwsze.
6. Operatory dodawania + i mnożenia * działają także na napisach, odpowiednio łącząc i replikując je.
7. Zdefiniowane przez użytkownika funkcje mogą przyjmować dowolnie wiele argumentów (nawet żadnego).
8. Argumenty pozycyjne są wprowadzane w wywołaniu funkcji w zdefiniowanej kolejności, argumenty nazwane po ich nazwie.
9. Argument domyślny posiada pewną predefiniowaną nazwę zawartą w definicji funkcji.

Wrocław University of Science and Technology

Rzeczy do zapamiętania!

10. Argumenty pozycyjne muszą poprzedzać argumenty nazwane.
11. Zmienne zdefiniowane poza funkcją mogą na czas działania funkcji zostać przykryte przez zmienne o tej samej nazwie (chyba, że poprzedzi je słowo kluczowe global), zmienne zdefiniowane w funkcji są zmiennymi lokalnymi i istnieją tylko w jej obrębie.
12. Słowo kluczowe return pozwala funkcji na zwracanie wartości (może być ich kilka, oddzielonych przecinkiem), jeśli słowo return nie występuje w ciele funkcji bądź po tym słowie nic się nie pojawia to funkcja zwraca brak wartości (None).
13. Zwracany wynik można łatwo przypisać do zmiennej, można tę operacje też pominąć.
14. Korzystając z funkcji print() wykorzystuj funkcję format() do formatowania liczb, np. do wyświetlenia odpowiedniego zaokrąglenia.