malgorzata.krawczyk(at)agh.edu.pl

konsultacje: e-mail

Zaliczenie

- Każde zajęcia laboratoryjne, oprócz pierwszych, rozpoczynają się krótkim sprawdzianem z materiału wcześniejszego, za które można uzyskać 25% punktów. Pozostałe 75% punktów można uzyskać za program napisany w czasie zajęć.
- Podstawowym terminem uzyskania zaliczenia jest koniec zajęć w danym semestrze. W przypadku braku zaliczenia w terminie podstawowym Student może dwukrotnie przystąpić do zaliczenia poprawkowego w formie ustnej.
- Ocena za dane zajęcia ustalana jest na podstawie ocen uzyskanych w czasie ich trwania.
- Przy obliczaniu oceny końcowej z laboratorium anulowana jest najsłabsza ocena ze sprawdzianu i najsłabsza ocena z programów.
- Ocena końcowa równa jest ocenie uzyskanej z ćwiczeń laboratoryjnych.
- Wysokość oceny końcowej będzie ustalana zgodnie ze skalą ocen obowiązującą w regulaminie AGH, przyporządkowującą procent opanowania materiału konkretnej ocenie (Par.13, pkt.1).

Organizacja zajęć laboratoryjnych - UPeL

- treść zadań
- przed końcem zajęć wszystkie pliki z danych zajęć należy wgrać do systemu, bez tego dane zajęcia nie będą oceniane
- czat (awaryjny)
- oceny + komentarze

Organizacja zajęć laboratoryjnych - MS Teams

- kanał dla całej grupy
- kanały dla poszczególnych osób (dana osoba+prowadzący)
- każda osoba musi mieć włączone w czasie zajęć oba kanały, przy czym pracuje i rozmawia z prowadzącym w swoim kanale, kanał grupowy pozwoli prowadzącemu na ewentualne zwrócenie się do wszystkich osób jednocześnie
- w kanale prywatnym Student przez cały czas trwania zajęć udostępnia cały pulpit oraz ma włączoną kamerę
- w czasie zajęć włączona może być tylko aplikacja Teams oraz przeglądarka, a w niej zakładki:
  - docs.python.org/3/reference/index.html
  - upel
  - repl.it
- w systemie repl.it proszę utworzyć nowy "repl" Python o nazwie ImieNazwisko i udostępnić go MKrawczyk, na początku każdych zajęć proszę usunąć całą zawartość katalogu, a przed ich końcem pobrać całą zawartość jako zip i zdeponować w systemie UPeL

Pierwszy program

hello.py

```
print('Hello')
```

Pierwszy program

hello.py

```
print('Hello')
```

$python3 hello.py

Pierwszy program

hello.py

```
print('Hello')
```

$python3 hello.py

albo

```
#!/usr/bin/env python3
print('Hello')
```

$chmod +x hello.py
$./hello.py

Uwaga

```
#!/usr/bin/env python3
print('Hello')
```

$python hello.py
Która wersja???

Uwaga

```
#!/usr/bin/env python3
print('Hello')
```

$python hello.py
Która wersja???

$python2 hello.py
Która wersja???

- komentarz jednowierszowy: #
- słowa kluczowe:

```
import keyword
keyword.kwlist
```

- komentarz jednowierszowy: #
- słowa kluczowe:

```
import keyword
keyword.kwlist
```

- info z poziomu skryptu/interpretera:

```
import math
dir(math)
help(math.modf)
```

- komentarz jednowierszowy: #
- słowa kluczowe:

```
import keyword
keyword.kwlist
```

- info z poziomu skryptu/interpretera:

```
import math
dir(math)
help(math.modf)

dir('')
help('').strip)
#''.strip.__doc__
```

- komentarz jednowierszowy: #
- słowa kluczowe:

```
import keyword
keyword.kwlist
```

- info z poziomu skryptu/interpretera:

```
import math
dir(math)
help(math.modf)

dir('')
help('').strip)
#''.strip.__doc__

type('')
type("")
```

Typy podstawowe

```
a=7
print(type(a))

a=1.5
print(type(a))
```

Typy podstawowe

```
a=7
print(type(a))

a=1.5
print(type(a))


a=1,1
print(type(a))
```

Typy podstawowe

```
a=7
print(type(a))

a=1.5
print(type(a))


a=1,1
print(type(a))

a,b=1,'2'
print(type(a), type(b))
```

Typy podstawowe

```
a=7
print(type(a))

a=1.5
print(type(a))


a=1,1
print(type(a))

a,b=1,'2'
print(type(a), type(b))

a,*b=1,'2',3.,4,5
print(type(a), type(b))
```

math

```
print(1/2, 1//2)
print(1./2, 1.//2)
```

math

```
print(1/2, 1//2)
print(1./2, 1.//2)


print(2**3, pow(2,3), math.pow(2,3))
print(pow(2,3,4), pow(2,3,5))
```

math

```
print(1/2, 1//2)
print(1./2, 1.//2)


print(2**3, pow(2,3), math.pow(2,3))
print(pow(2,3,4), pow(2,3,5))


print(math.ceil(1/3), math.floor(1/3), round(1/3), round(1/3,3))
print(math.ceil(2/3), math.floor(2/3), round(2/3), round(2/3,3))
```

math

```
print(1/2, 1//2)
print(1./2, 1.//2)


print(2**3, pow(2,3), math.pow(2,3))
print(pow(2,3,4), pow(2,3,5))


print(math.ceil(1/3), math.floor(1/3), round(1/3), round(1/3,3))
print(math.ceil(2/3), math.floor(2/3), round(2/3), round(2/3,3))


print(math.modf(1/3), math.modf(2.5))
```

math

```
print(1/2, 1//2)
print(1./2, 1.//2)


print(2**3, pow(2,3), math.pow(2,3))
print(pow(2,3,4), pow(2,3,5))


print(math.ceil(1/3), math.floor(1/3), round(1/3), round(1/3,3))
print(math.ceil(2/3), math.floor(2/3), round(2/3), round(2/3,3))


print(math.modf(1/3), math.modf(2.5))


print(min(2,11,3,4,2),max(2,11,3,4,2))
```

math

```
print(1/2, 1//2)
print(1./2, 1.//2)


print(2**3, pow(2,3), math.pow(2,3))
print(pow(2,3,4), pow(2,3,5))


print(math.ceil(1/3), math.floor(1/3), round(1/3), round(1/3,3))
print(math.ceil(2/3), math.floor(2/3), round(2/3), round(2/3,3))


print(math.modf(1/3), math.modf(2.5))


print(min(2,11,3,4,2),max(2,11,3,4,2))


a=-1.7
print(abs(a), math.fabs(a))
a=-1
print(abs(a), math.fabs(a))
```

Instrukcja warunkowa

```
if warunek1:
 pass
elif warunek2:
 pass
else:
 pass
```

Instrukcja warunkowa

```
if warunek1:
 pass
elif warunek2:
 pass
else:
 pass
```

Zadanie:
proszę napisać program umożliwiający rozwiązanie równania kwadratowego

Instrukcja warunkowa

```
from math import sqrt
from cmath import sqrt as csqrt

a=float(input('a=?'))
b=float(input('b=?'))
c=float(input('c=?'))

d=b*b-4*a*c
if d>1e-6:
 x1=(-b-sqrt(d))/(2*a)
 x2=(-b+sqrt(d))/2/a
 print(f'x1={x1:.3f}, x2={x2:.3f}')
elif abs(d)<=1e-6:
 x=-b/(2*a)
 print(f'x1=x2={x}')
else:
 x1=(-b-csqrt(d))/(2*a)
 x2=(-b+csqrt(d))/(2*a)
 print(f'x1={x1:.3f}, x2={x2:.3f}')
```

Instrukcja warunkowa

```
import sys
import math
import cmath

if len(sys.argv)!=5:
 sys.exit()

a=float(sys.argv[1])
b=float(sys.argv[2])
c=float(sys.argv[3])
eps=float(sys.argv[4])

if (d:=b**2-4*a*c)>eps:
 x1=(-b-math.sqrt(d))/(2*a)
 x2=(-b+math.sqrt(d))/(2*a)
 print(f'{x1=:.3f}, {x2=:.3f}')
elif math.fabs(d)<=eps:
 print(f'x1=x2={-b/(2*a):.2f}')
else:
 x1=(-b-cmath.sqrt(d))/(2*a)
 x2=(-b+cmath.sqrt(d))/(2*a)
 print(f'{x1=:.3f}, {x2=:.3f}')

print(d)
```

Krotki

```
k=()
print(type(k))
```

```
k=(2)
print(type(k))
```

```
k=(2,)
print(type(k))
```

Krotki

```
k=()
print(type(k))
```

```
k=(2)
print(type(k))
```

```
k=(2,)
print(type(k))
```

```
k=(1,2.3,'3',(4,7),[2,3,4],)
print(len(k))
```

Krotki

```
k=()
print(type(k))
```

```
k=(2)
print(type(k))
```

```
k=(2,)
print(type(k))
```

```
k=(1,2.3,'3',(4,7),[2,3,4],)
print(len(k))
```

```
print(k[0], k[len(k)-1], k[-1])
```

Krotki

```
k=()
print(type(k))
```

```
k=(2)
print(type(k))
```

```
k=(2,)
print(type(k))
```

```
k=(1,2.3,'3',(4,7),[2,3,4],)
print(len(k))
```

```
print(k[0], k[len(k)-1], k[-1])
```

```
#k[-1]='a'
k[-1][1]='a'
```

Listy

```
k=[]
print(type(k))
```

```
k=[2]
print(type(k))
```

```
k=[2,]
print(type(k))
```

Listy

```
k=[]
print(type(k))
```

```
k=[2]
print(type(k))
```

```
k=[2,]
print(type(k))
```

```
k=[1,2.3,'3',(4,7),[2,3,4],]
print(len(k))
```

Listy

```
k=[]
print(type(k))
```

```
k=[2]
print(type(k))
```

```
k=[2,]
print(type(k))
```

```
k=[1,2.3,'3',(4,7),[2,3,4],]
print(len(k))
```

```
print(k[0], k[len(k)-1], k[-1])
```

Listy

```
k=[]
print(type(k))
```

```
k=[2]
print(type(k))
```

```
k=[2,]
print(type(k))
```

```
k=[1,2.3,'3',(4,7),[2,3,4],]
print(len(k))
```

```
print(k[0], k[len(k)-1], k[-1])
```

```
#k[-2][1]='a'
k[-2]='a'
```

Uwaga

```
bool(0), bool(1)          #(False, True)

bool([]), bool([1])       #(False, True)

bool(''), bool('a')       #(False, True)

a,b=1,None
bool(a), bool(b)          #(True, False)
```

Wycinki list

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
```

Wycinki list

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

print(k[:])
```

Wycinki list

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

print(k[:])

print(k[2:-3])
```

Wycinki list

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

print(k[:])

print(k[2:-3])

print(k[2:-3:2])
```

Wycinki list

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
print(k[:])
print(k[2:-3])
print(k[2:-3:2])
print(k[2:])
```

Wycinki list

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
print(k[:])
print(k[2:-3])
print(k[2:-3:2])
print(k[2:])
print(k[:-3])
```

Wycinki list

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
print(k[:])
print(k[2:-3])
print(k[2:-3:2])
print(k[2:])
print(k[:-3])
print(k[::-1])
```

Kopiowanie list

```
k=[1,2.3,'3',(4,7),[2,3,4],]

c=k
c[1]=[7,8,9]
print(c,k)
print(id(c),id(k))
```

Kopiowanie list

```
k=[1,2.3,'3',(4,7),[2,3,4],]

c=k
c[1]=[7,8,9]
print(c,k)
print(id(c),id(k))


c=k[:]
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))
```

Kopiowanie list

```
k=[1,2.3,'3',(4,7),[2,3,4],]

c=k
c[1]=[7,8,9]
print(c,k)
print(id(c),id(k))


c=k[:]
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))

c[-1][1]='7,8,9'
print(c,k)
```

Kopiowanie list

```
k=[1,2.3,'3',(4,7),[2,3,4],]

c=k
c[1]=[7,8,9]
print(c,k)
print(id(c),id(k))


c=k[:]
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))

c[-1][1]='7,8,9'
print(c,k)


c=k.copy()
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))
```

Kopiowanie list

```
k=[1,2.3,'3',(4,7),[2,3,4],]

c=k
c[1]=[7,8,9]
print(c,k)
print(id(c),id(k))


c=k[:]
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))

c[-1][1]='7,8,9'
print(c,k)


c=k.copy()
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))

c[-1][1]='7,8,9'
print(c,k)
```

Kopiowanie list

```
import copy

k=[1,2.3,'3',(4,7),[2,3,4],]

c=copy.copy(k)
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))
```

Kopiowanie list

```
import copy

k=[1,2.3,'3',(4,7),[2,3,4],]

c=copy.copy(k)
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))


c[-1][1]='7,8,9'
print(c,k)
```

Kopiowanie list

```
import copy

k=[1,2.3,'3',(4,7),[2,3,4],]

c=copy.copy(k)
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))


c[-1][1]='7,8,9'
print(c,k)


c=copy.deepcopy(k)
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))
```

Kopiowanie list

```
import copy

k=[1,2.3,'3',(4,7),[2,3,4],]

c=copy.copy(k)
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))


c[-1][1]='7,8,9'
print(c,k)


c=copy.deepcopy(k)
c[1]='7,8,9'
print(c,k)
print(id(c),id(k))


c[-1][1]='7,8,9'
print(c,k)
```

Zliczanie/znajdowanie elementu w liście

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

print(k.count(13))
print(k.count(-13))
```

Zliczanie/znajdowanie elementu w liście

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

print(k.count(13))
print(k.count(-13))


print(k.index(13))
#print(k.index(-13))
```

Zliczanie/znajdowanie elementu w liście

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

print(k.count(13))
print(k.count(-13))


print(k.index(13))
#print(k.index(-13))



print(13 in k)
print(13 not in k)
```

Zliczanie/znajdowanie elementu w liście

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

print(k.count(13))
print(k.count(-13))


print(k.index(13))
#print(k.index(-13))



print(13 in k)
print(13 not in k)


if 13 in k:
 pass

if 13 not in k:
 pass
```

Wstawianie elementu do listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

k.insert(4,-13)
print(k)
```

Wstawianie elementu do listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

k.insert(4,-13)
print(k)

k.insert(-23,4)
k.insert(23,4)
print(k)
```

Wstawianie elementu do listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

k.insert(4,-13)
print(k)

k.insert(-23,4)
k.insert(23,4)
print(k)


k[1:4]=[7,8,9,10,]          #(7,8,9,10,), '78'
print(k)

k[1:4]=[[7,8,],]
print(k)
```

Usuwanie elementu/ów listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]


k.remove(1)                    #k.remove(-4)
print(k)
```

Usuwanie elementu/ów listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]


k.remove(1)                    #k.remove(-4)
print(k)


del k[3]                       #del k[-23]
print(k)
```

Usuwanie elementu/ów listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]


k.remove(1)                    #k.remove(-4)
print(k)


del k[3]                       #del k[-23]
print(k)

del k[-3:]
print(k)
```

Usuwanie elementu/ów listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]


k.remove(1)                    #k.remove(-4)
print(k)


del k[3]                       #del k[-23]
print(k)

del k[-3:]
print(k)


print(k.pop())
print(k)

print(k.pop(-3))               #print(k.pop(-23))
print(k)
```

Usuwanie elementu/ów listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]


k.remove(1)                    #k.remove(-4)
print(k)


del k[3]                       #del k[-23]
print(k)

del k[-3:]
print(k)


print(k.pop())
print(k)

print(k.pop(-3))               #print(k.pop(-23))
print(k)


k.clear()
print(k)
```

Rozbudowywanie listy

```
k=[1]*10
k[3]+=1
```

Rozbudowywanie listy

```
k=[1]*10
k[3]+=1
```

```
k=[[]]*10
k[3].append(1)
```

Rozbudowywanie listy

```
k=[1]*10
k[3]+=1
```

```
k=[[]]*10
k[3].append(1)
```

```
k=[[] for i in range(10)]
k[3].append(1)
```

Rozbudowywanie listy

```
k=[1]*10
k[3]+=1
```

```
k=[[]]*10
k[3].append(1)
```

```
k=[[] for i in range(10)]
k[3].append(1)
```

```
k[3].append([1,2,3])
print(k)
k[3].extend([1,2,3])
print(k)
```

Range

```
k=[]
for i in range(10):
    k.append(i)
```

Range

```
k=[]
for i in range(10):
    k.append(i)
```

```
k=list(range(10))
```

Range

```
k=[]
for i in range(10):
    k.append(i)
```

```
k=list(range(10))
```

```
k=list(range(3,10))
```

Range

```
k=[]
for i in range(10):
  k.append(i)
```

```
k=list(range(10))
```

```
k=list(range(3,10))
```

```
k=list(range(3,10,2))
```

Range

```
k=[]
for i in range(10):
  k.append(i)
```

```
k=list(range(10))
```

```
k=list(range(3,10))
```

```
k=list(range(3,10,2))
```

```
k=list(range(10,0,-1))
```

Range

```
k=[]
for i in range(10):
  k.append(i)
```

```
k=list(range(10))
```

```
k=list(range(3,10))
```

```
k=list(range(3,10,2))
```

```
k=list(range(10,0,-1))
```

```
k=[i for i in range(10)]
```

Pętla

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
for i in k:
    i*=2
    print(i, end=', ')

print('\n',k)
```

Pętla

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
for i in k:
    i*=2
    print(i, end=', ')

print('\n',k)


for i in range(len(k)):
    k[i]*=2
```

Pętla

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
for i in k:
    i*=2
    print(i, end=', ')

print('\n',k)


for i in range(len(k)):
    k[i]*=2


for i,v in enumerate(k):
    k[i]=1 if v>0 else -1
```

Pętla

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
for i in k:
    i*=2
    print(i, end=', ')

print('\n',k)


for i in range(len(k)):
    k[i]*=2


for i,v in enumerate(k):
    k[i]=1 if v>0 else -1


for i in k:
    if i%2:          #if not i%2:
        break
else:
    print('kiedy?')
```

Lista składana

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

np=[i for i in k if i%2]

np=[1 if i>0 else -1 for i in k]
```

Lista składana

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

np=[i for i in k if i%2]

np=[1 if i>0 else -1 for i in k]
```

```
k=[(k[i],k[-i-1]) for i in range(len(k)//2)]
print(k)
```

Lista składana

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]

np=[i for i in k if i%2]

np=[1 if i>0 else -1 for i in k]
```

```
k=[(k[i],k[-i-1]) for i in range(len(k)//2)]
print(k)
```

```
for i,j in k:
  print(i,j)
```

Sortowanie

```
k=[(89, 34), (92, 31), (96, 0), (48, 30), (38, 10),]

c=k[:]
c.sort()          #c=c.sort()
print(c)
```

Sortowanie

```
k=[(89, 34), (92, 31), (96, 0), (48, 30), (38, 10),]

c=k[:]
c.sort()          #c=c.sort()
print(c)


c=k[:]
c.sort(key=lambda x: x[1])
print(c)
```

Sortowanie

```
k=[(89, 34), (92, 31), (96, 0), (48, 30), (38, 10),]

c=k[:]
c.sort()          #c=c.sort()
print(c)


c=k[:]
c.sort(key=lambda x: x[1])
print(c)


c=k[:]
c.sort(reverse=True)
print(c)
```

Sortowanie

```
k=[(89, 34), (92, 31), (96, 0), (48, 30), (38, 10),]

c=k[:]
c.sort()          #c=c.sort()
print(c)


c=k[:]
c.sort(key=lambda x: x[1])
print(c)


c=k[:]
c.sort(reverse=True)
print(c)


c=k[:]
for i,j in sorted(k):              #for i in sorted(k):
  print(i,j)                       #  print(i[0],i[1])
print(c)
```

Odwrócenie listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
```

```
c=k[:]
c.sort(reverse=True)
print(c)
```

Odwrócenie listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
```

```
c=k[:]
c.sort(reverse=True)
print(c)
```

```
c=k[:]
c.reverse()
print(c)
```

Odwrócenie listy

```
k=[8, 0, 17, 1, 10, 13, 19, 13, 10, 3,]
```

```
c=k[:]
c.sort(reverse=True)
print(c)
```

```
c=k[:]
c.reverse()
print(c)
```

```
c=k[::-1]
print(c)
```

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy
2. j.w. ale korzystając z pętli *while*

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy
2. j.w. ale korzystając z pętli *while*
3. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy począwszy od elementu o indeksie 1 (bez instrukcji warunkowej)

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy
2. j.w. ale korzystając z pętli *while*
3. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy począwszy od elementu o indeksie 1 (bez instrukcji warunkowej)
4. j.w. ale bez *range*

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy
2. j.w. ale korzystając z pętli *while*
3. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy począwszy od elementu o indeksie 1 (bez instrukcji warunkowej)
4. j.w. ale bez *range*
5. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy od końca (bez instrukcji warunkowej)

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy
2. j.w. ale korzystając z pętli *while*
3. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy począwszy od elementu o indeksie 1 (bez instrukcji warunkowej)
4. j.w. ale bez *range*
5. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy od końca (bez instrukcji warunkowej)
6. j.w. ale bez *range*

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy
2. j.w. ale korzystając z pętli *while*
3. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy począwszy od elementu o indeksie 1 (bez instrukcji warunkowej)
4. j.w. ale bez *range*
5. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy od końca (bez instrukcji warunkowej)
6. j.w. ale bez *range*
7. proszę utworzyć nową listę, której elementami są krotki (index, element) na podstawie istniejącej listy $\leftarrow$ lista składana

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy
2. j.w. ale korzystając z pętli *while*
3. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy począwszy od elementu o indeksie 1 (bez instrukcji warunkowej)
4. j.w. ale bez *range*
5. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy od końca (bez instrukcji warunkowej)
6. j.w. ale bez *range*
7. proszę utworzyć nową listę, której elementami są krotki (index, element) na podstawie istniejącej listy $\leftarrow$ lista składana
8. proszę posortować listę z poprzedniego punktu wg drugiego elementu krotek

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy
2. j.w. ale korzystając z pętli *while*
3. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy począwszy od elementu o indeksie 1 (bez instrukcji warunkowej)
4. j.w. ale bez *range*
5. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy od końca (bez instrukcji warunkowej)
6. j.w. ale bez *range*
7. proszę utworzyć nową listę, której elementami są krotki (index, element) na podstawie istniejącej listy ← lista składana
8. proszę posortować listę z poprzedniego punktu wg drugiego elementu krotek
9. proszę utworzyć nową listę, której elementami są krotki (index, element) na podstawie istniejącej listy, przy czym dodajemy krotkę tylko, jeśli wartość pobrana z listy jest wartością parzystą ← lista składana

Zadania

1. korzystając z pętli *for* proszę usunać wszystkie wystąpienia określonej wartości z listy
2. j.w. ale korzystając z pętli *while*
3. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy począwszy od elementu o indeksie 1 (bez instrukcji warunkowej)
4. j.w. ale bez *range*
5. korzystając z pętli *for* oraz funkcji *range* proszę wypisać co drugi element listy od końca (bez instrukcji warunkowej)
6. j.w. ale bez *range*
7. proszę utworzyć nową listę, której elementami są krotki (index, element) na podstawie istniejącej listy ← lista składana
8. proszę posortować listę z poprzedniego punktu wg drugiego elementu krotek
9. proszę utworzyć nową listę, której elementami są krotki (index, element) na podstawie istniejącej listy, przy czym dodajemy krotkę tylko, jeśli wartość pobrana z listy jest wartością parzystą ← lista składana
10. proszę utworzyć nową listę, której elementami są krotki (index, element) lub (element, index) na podstawie istniejącej listy, tak, aby pierwszy element krotki był mniejszy od drugiego ← lista składana