malgorzata.krawczyk(at)agh.edu.pl

konsultacje: D7/311, e-mail

Zaliczenie

- Każde zajęcia laboratoryjne, oprócz pierwszych, rozpoczynają się krótkim sprawdzianem z materiału wcześniejszego, za który 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 jest ustalana zgodnie ze skalą ocen obowiązującą w regulaminie AGH, przyporządkowującą procent opanowania materiału konkretnej ocenie (Par.13, pkt.1).

Organizacja zajęć laboratoryjnych - UPeL

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

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 python3.12
print('Hello')
```

$python3 hello.py
Która wersja???

Uwaga

```
#!/usr/bin/env python3.12
print('Hello')
```

$python3 hello.py
Która wersja???

$python3.11 hello.py
Która wersja???

- komentarz jednowierszowy: #
- słowa kluczowe:

```
import keyword
keyword.kwlist
```

- komentarz jednowierszowy: #
- słowa kluczowe:

```
import keyword
keyword.kwlist
```

- funkcje wbudowane:

```
import builtins
dir(builtins)
```

- komentarz jednowierszowy: #
- słowa kluczowe:

```
import keyword
keyword.kwlist
```

- funkcje wbudowane:

```
import builtins
dir(builtins)
```

- info z poziomu skryptu/interpretera:

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

- komentarz jednowierszowy: #
- słowa kluczowe:

```
import keyword
keyword.kwlist
```

- funkcje wbudowane:

```
import builtins
dir(builtins)
```

- 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
```

- funkcje wbudowane:

```
import builtins
dir(builtins)
```

- 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
```

```
a=w1 if warunek else w2
```

Instrukcja warunkowa

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

```
a=w1 if warunek else w2
```

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])
```