Elágazás

2025

1 Logikai és összehasonlító műveletek

1.0.1 Logikai műveletek

Logikai értékek közti műveleteket valósítanak meg a Boole-algebra alapján.

A	**B**	A and B	A or B	not A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

[ ]:
```
# Logikai változók létrehozása
A = True
B = False

# ÉS (and)
logikai_es = A and B
print("Logikai ÉS:", logikai_es)

# VAGY (or)
logikai_vagy = A or B
print("Logikai VAGY:", logikai_vagy)

# NEM (not)
logikai_nem = not A
print("Logikai NEM:", logikai_nem)
```

A Python alapvetően nem rendelkezik beépített kizáró vagy (xor) operátorral, de a következő kifejezés ugyanazt az eredményt adja: (az xor akkor igaz, ha csak az egyik bemenet igaz) | **A** | **B** | A xor B | |:——:|:——:|:——-:| | False | False | False | | False | True | True | | True | False | True | | True | True | False |

[ ]:
```
A = True
B = False

kifejezes = (A and not B) or (not A and B)
print(kifejezes)  # eredmény: True
```

A De Morgan-azonosságokat logikailag a következőképpen fejezhetjük ki: (később szükség lesz rá)

```
nem (a és b) = (nem a) vagy (nem b)
nem (a vagy b) = (nem a) és (nem b)
```

1.0.2 Összehasonlító műveletek

Azonos típusú elemeket tudunk összehasonlítani, amelynek eredménye logikai változó lesz.

[ ]:
```
# Összehasonlításra használható azonos típusú változók
a = 5
b = 3

# Egyenlőség (==) (a értéke 5?)
egyenloseg = a == 5
print("Egyenlőség:", egyenloseg)

# Nem egyenlőség (!=) (a értéke nem egyenlő b értékével?)
nem_egyenloseg = a != b
print("Nem egyenlőség:", nem_egyenloseg)

# Nagyobb/magasabb (>), kisebb/alacsonyabb (<) (a nagyobb, mint b), (a kisebb,␣
 ↪mint 7)
nagyobb = a > b
kisebb = a < 7
print("Nagyobb:", nagyobb)
print("Kisebb:", kisebb)

# Nagyobb vagy egyenlő (>=), kisebb vagy egyenlő (<=) (a nagyobb vagy egyenlő,␣
 ↪mint 5), (a kisebb vagy egyenlő, mint b)
nagyobb_vagy_egyenlo = a >= 5
kisebb_vagy_egyenlo = a <= b
print("Nagyobb vagy egyenlő:", nagyobb_vagy_egyenlo)
print("Kisebb vagy egyenlő:", kisebb_vagy_egyenlo)
```

2 Elágazás

Az elágazások a vezérlési szerkezetek egyik alapvető formáját képezik a programozásban. Pythonban az elágazások az `if`, `elif` (opcionális), és `else` kulcsszavakkal valósulnak meg.

2.1 if szerkezet

Az `if` kulcsszót egy kifejezés követ, amely logikai igaz vagy hamis értéket ad. A kifejezés tartalmazhat összehasonlító műveleteket (==, !=, >, >=, <, <=) és logikai kapcsolatokat (not, and, or).

Ha a kifejezés igaz értéket ad, akkor a mögötte lévő blokk le fog futni.

A Pythonban a blokkokat indentációval (behúzással) jelöljük. Ez lehet kettő vagy négy space. Az

elágazásoknál a beljebb helyezett blokkok az `if`, `elif`, és `else` kulcsszavak alatt találhatók. A Python szigorúan veszi az indentációt, és a helytelen behúzás hibát eredményez.

```
if <feltétel> :
    utasítások
```

[ ]:
```
a = 5
b = 20

# Ha b nagyobb, mint a, akkor írja ki, hogy "'b' nagyobb, mint 'a'"
if b > a:
    print("igaz az elágazás, mert")
    print("'b' nagyobb, mint 'a'")

print("következő utasítás")
```

2.2 else szerkezet

Az `else` kulcsszó segítségével megadhatunk egy olyan esetet, amely csak akkor fog érvényesülni, ha az előtte lévő feltétel nem teljesült, így nem kell azt külön egy másik `if` feltétellel megvizsgálni.

```
if <feltétel> :
    utasítások
else :
    utasítások
```

[ ]:
```
a = 10
b = 20

# Ha b nagyobb, mint a, akkor írja ki, hogy "'b' nagyobb, mint 'a'",
# különben írja ki, hogy "'b' nem nagyobb, mint 'a'"
if b > a:
    print("'b' nagyobb, mint 'a'")
else:
    print("'b' nem nagyobb, mint 'a'")

print("következő utasítás")
```

2.3 Egymásba ágyazás

Az egyes feltételek egymásba ágyazhatók, azonban figyeli kell arra, hogy az egyes elágazásokhoz mekkora behúzás tartozik.

[ ]:
```
a = 20
b = 20

# Ha b nagyobb, mint a, akkor írja ki, hogy "'b' nagyobb, mint 'a'",
# de ha b kisebb, mint a, akkor írja ki, hogy "'b' kisebb, mint 'a'",
# különben írja ki, hogy "'b' ugyanakkora, mint 'a'"
```

```
if b > a:
    print("'b' nagyobb, mint 'a'")
else:
    if b < a:
        print("'b' kisebb, mint 'a'")
    else:
        print("'b' ugyanakkora, mint 'a'")

print("következő utasítás")
```

2.4 elif szerkezet

Az `elif` kulcsszó segítségével egy újabb feltételt vizsgálhatunk meg, ha az előző feltétel nem teljesült (a fenti példában szereplő `else: if` szerkezet összevonása). Így több esetet fel lehet sorolni egymás után. Az `else` ág csak akkor fog lefutni, ha az előtte található egyik feltétel sem volt igaz.

```
if <feltétel> :
    utasítások
elif <feltétel> :
    utasítások
elif <feltétel> :
    utasítások
else :
    utasítások
```

[ ]:
```
a = 20
b = 20

# Ha b nagyobb, mint a, akkor írja ki, hogy "'b' nagyobb, mint 'a'",
# de ha b kisebb, mint a, akkor írja ki, hogy "'b' kisebb, mint 'a'",
# különben írja ki, hogy "'b' ugyanakkora, mint 'a'"
if b > a:
    print("'b' nagyobb, mint 'a'")
elif b < a:
    print("'b' kisebb, mint 'a'")
else:
    print("'b' ugyanakkora, mint 'a'")

print("következő utasítás")
```

2.5 Rövidített if elágazás

Ha csak egyetlen utasítást szeretnénk végrehajtani az elágazásban, akkor azt le lehet rövidíteni.

[ ]:
```
a = 25
b = 20

if a > b: print("'a' nagyobb, mint 'b'")
```

Ugyancsak létezik rövidítés az `if-else` szerkezet egyszerűsített használatára is. (`[a] if <condition> else [b]`)
Más nyelvekben ezt a szerkezetet feltételes kifejezésként, vagy háromtagú operátorként ismerhetjük.

[ ]:
```
a = 25
b = 20

# melyik a nagyobb
nagyobb = a if a > b else b
print("Nagyobb:", nagyobb)
# erre van beépített függvény
# print("Maximum:", max(a,b)) #figyeljünk arra, hogy ne legyen max nevű␣
 ↪változó, mert akkor már nem hívható a max() fgv.
```

2.6 Feltételek összekapcsolása

Két feltétel együttes teljesülését logikai műveletekkel össze lehet kapcsolni.

[ ]:
```
a = 99

# 'a' kétjegyű?
if a >= 10:
    if a < 100:
        print("'a' kétjegyű")

# logikai 'és' kapcsolattal
if a > 9 and a < 100:
    print("'a' kétjegyű")
```

Logikai ‘és’ kapcsolattal rendelkező tartomány felírása tovább egyszerűsíthető:

[ ]:
```
a = 99

# tartomány esetén a logikai 'és' kapcsolat felírható így is
if 9 < a < 100:
    print("'a' kétjegyű")
```

2.7 Hibás kód

Ha egy változót egy `if` szerkezetben hozunk létre, akkor előfordulhat olyan eset, mikor az nem fog létrejönni. Ha a későbbiekben hivatkozunk arra a változóra, akkor hibát fogunk kapni.
Vagy készítsük el az összes esetet, vagy hozzuk létre a változót egy alapértelmezett értékkel.
A `None` használatával jelezhetjük, hogy egy változó még nem lett inicializálva vagy beállítva, és ezt a későbbiekben fogjuk megtenni. pl.: `szoveg = None`

[ ]:
```
ev = 1

if ev > 6:
    szoveg = "Iskoláskorú"
```

```
# Ez hibát fog okozni!
# Nem inicializálta a szoveg változót mindkét ágon
print(szoveg)
```

2.8 match-case szerkezet

A Python 3.10 verziójától kezdve bevezetésre került az `match-case` szerkezet, amely egy kifejezés alapján történő mintaillesztésre szolgál. Ez a funkcionalitás egyfajta alternatívája a hagyományos `switch`-nek.
Az átadott kifejezést hasonlítja össze a `case` értékkel, az első megtalált mintában szereplő utasításokat végrehajtja, majd kilép a szerkezetből.
Ha nincs egyetlen illeszkedő minta sem, akkor azt a `_` alapértelmezett ággal lehet lekezelni, amit a legutolsó helyen kell létrehozni.

[ ]:
```
kifejezes = 5

match kifejezes:
    case 1 | 5:
        print("Első ág")
    case 2:
        print("Második ág")
    case 3:
        print("Harmadik ág")
    case _:
        print("Alapértelmezett ág")
```

3 Gyakorló feladatok

1. Feladat

   Döntse el egy számról, hogy páros vagy páratlan!

[ ]:
```
szam = 7

eredmeny = "Páros" if szam % 2 == 0 else "Páratlan"
print(eredmeny)
```

2. Feladat

   Döntse el egy számról, hogy 4-el osztható-e de 5-el nem!

[ ]:
```
szam = 20

if szam % 4 == 0 and szam % 5 != 0:
    print("4-el osztható-e de 5-el nem!")
elif szam % 4 == 0:
    print("csak 4-el osztható")
```

3. Feladat

   Adjon meg egy érdemjegyet (0-5), majd írja ki szövegesen (nem teljesített, elégtelen, elégséges, közepes, jó, jeles). Ha olyan értéket adunk meg, amely nincs a tartományban, akkor írja ki, hogy 'érvénytelen jegy'.

[ ]:
```
jegy = 5

if jegy == 0:
    print("Nem teljesített")
elif jegy == 1:
    print("Elégtelen")
elif jegy == 2:
    print("Elégséges")
elif jegy == 3:
    print("Közepes")
elif jegy == 4:
    print("Jó")
elif jegy == 5:
    print("Jeles")
else:
    print("Érvénytelen jegy")
```

4. Feladat

   Határozza meg, hogy egy ZH hogy sikerült (elégtelen, elégséges, közepes, jó, jeles), ha a ZH maximum 60 pontos volt. A jegyet a BME-n alkalmazott százalékok alapján határozza meg.

[ ]:
```
zh = 65

if zh < 40:
    print("Elégtelen")
elif 40 <= zh < 55:
    print("Elégséges")
elif 55 <= zh < 70:
    print("Közepes")
elif 70 <= zh < 85:
    print("Jó")
elif zh >= 85:
    print("Jeles")
else:
    print("Érvénytelen jegy")
```

5. Feladat

   Egy üzletben nagy leárazások vannak. Kinéztünk egy kabátot, a címkéjén az ár 38 990 Ft. A kabátra 20%-os árengedményt adnak, amelyet a pénztárnál fognak érvényesíteni. Mennyit kell adnunk, ha készpénzben szeretnénk fizetni? (5 Ft-ra kell kerekíteni)

[ ]:
```
eredeti_ar = 38991
kedvezmeny = 0.20
uj_ar = eredeti_ar * (1 - kedvezmeny)
maradek = uj_ar % 5
if maradek > 2:
    uj_ar += 5 - maradek
else:
    uj_ar -= maradek
print(round(uj_ar), "Ft a kedvezményes ár.")
```

6. Feladat

   Készítsen egy programot, amely meghatározza az ‘a, b, c’ paraméterrel rendelkező másodfokú függvény esetén, hogy mi a megoldás. Helyettesítsen vissza és ellenőrizze le.

[ ]:
```
import math
import cmath

a, b, c = 1, 2, 1

diszkrininans = b**2 - 4 * a * c

if diszkrininans > 0:
    x1 = (-b + math.sqrt(diszkrininans)) / (2 * a)
    x2 = (-b - math.sqrt(diszkrininans)) / (2 * a)
    print("Két valós megoldás:")
    print(f"x1: {x1:.3f}")
    print(f"x2: {x2:.3f}")
    # ellenőrzés
    ell1 = a * x1**2 + b * x1 + c
    ell2 = a * x2**2 + b * x2 + c
    if math.fabs(ell1) < 1e-6 and math.fabs(ell2) < 1e-6:
        print("Ellenőrzés: OK")
elif diszkrininans == 0:
    x1 = (-b) / (2 * a)
    print("Egy valós megoldás:")
    print(f"x1: {x1:.3f}")
else:
    x1 = (-b + cmath.sqrt(diszkrininans)) / (2 * a)
    x2 = (-b - cmath.sqrt(diszkrininans)) / (2 * a)
    print("Komplex konjugált:")
    print(f"x1: {x1:.3f}")
    print(f"x2: {x2:.3f}")
```

7. Feladat

   Kérje be, hogy hány autó áll egy 100 hellyel rendelkező parkolóba. Írja ki, hogy még hány szabad hely van. Ha nincs több hely, akkor írja ki, hogy “Tele”.

[ ]:
```python
autok_szama = 30

if autok_szama < 100:
    print("Szabad helyek szama:", 100 - autok_szama)
else:
    print("Tele a parkoló")
```

8. Feladat

   Adjon meg az életkorát. Írja ki, hogy melyik korosztály (`Kiskorú`,`Felnőtt`,`Idős`). `Felnőtt` a 18 és 65 év közötti személy.

[ ]:
```python
eletkor = int(input("Kérem, adja meg az életkorát: "))

if eletkor < 18:
    print("Kiskorú")
elif 18 <= eletkor <= 65:
    print("Felnőtt")
else:
    print("Idős")
```

9. Feladat

   Határozzuk meg, hogy egy év szökőév vagy sem. (Az X év szökőév ha X osztható 4-gyel. Viszont nem az, ha X többszöröse 100-nak, kivéve, ha X 400-nak többszöröse).

[ ]:
```python
ev = 2024

if (ev % 4 == 0 and ev % 100 != 0) or (ev % 400 == 0):
    print(f"{ev} szökőév.")
else:
    print(f"{ev} nem szökőév.")
```