Alkalmi Python - 8. rész

PCLinuxOS Magazine - 2109. szeptember

critter (Peter Kelly)

Analóg óra, második rész

Az analóg óránk végső két opciója az arab, vagy római számok hozzáadása. Az órát egy lapra rajzoltuk fel a körvonalhoz tollat, és a kitöltéshez ecsetet használva. A festés pontos helyét egy ponthalmaz adja meg, meghatározva a körvonalak koordinátáit. Például a pontokat x és y középpontú kis és nagy átlójú ellipszisként rajzoltuk meg, ahogy egy kört szokás. A mutatók kezdő és vég x-y koordinátái adottak, a szélességüket a toll vastagsága adja meg. A szöveg felírásához meg kell határozni a szöveg színét és a betűk méretét, amit én 12-re állítottam. A beállítások érvényesítéséhez ezután a Qpainter setfont eljárást hívjuk meg.

A kiinduló pontunk az óra számlapjának közepe, ezért azzal annak a helynek a középre kell mennünk, vagy „átlépnünk”, ahová a szöveget szánjuk. A szöveg megrajzolásához a beillesztési ponthoz képest a betűméret felével balra és fölfelé elcsúsztatjuk. Esetünkben ez -6 és 6, ezért készítünk egy Qpoint objektumot ezekkel a koordinátákkal. Ha most ezt a Qpoint-ot és a szöveget a megrajzoláshoz a Qpainter drawText eljárásának átadjuk, a szöveg pontosan megrajzolódik.

A szöveg kiírásával kapcsolatban az óralap középpontjától mérve 75-ös sugár mellett döntöttem. A Pythonra is bízhatnánk a pontok kiszámítását, de mivel az óralapot másodpercenként 10-szer rajzoljuk fel ez sok számolást jelentene. Szerencsére a számlap eléggé szimmetrikus, ezért a számok elég egyszerűek lesznek (csak 0, 37, 65 és 75 a 75-ös sugárból 30 fokos lépésekkel számolva), így egyszerűen begépelhetjük a kódba.

Ha már a számokat felírtuk, visszamehetünk középre, készen a következő szám felrajzolására. Arab számok esetén ez valahogy így néz ki:

```
if numerals == 'arabic':
font.setPointSize(12)
qp.setFont(font)
p = QPoint(-6, 6)
qp.translate(37, -65) # move out
qp.drawText(p, '1') # paint the number
qp.translate(-37, 65) # move back

qp.translate(65, -37)
qp.drawText(p, '2')
qp.translate(-65, 37)

qp.translate(75, 0)
qp.drawText(p, '3')
qp.translate(-75, 0)

qp.translate(65, 37)
qp.drawText(p, '4')
qp.translate(-65, -37)

qp.translate(37, 65)
qp.drawText(p, '5')
qp.translate(-37, -65)

qp.translate(0, 75)
qp.drawText(p, '6')
qp.translate(0, -75)

qp.translate(-37, 65)
qp.drawText(p, '7')
qp.translate(37, -65)

qp.translate(-65, 37)
qp.drawText(p, '8')
qp.translate(65, -37)

qp.translate(-75, 0)
qp.drawText(p, '9')
qp.translate(75, 0)

qp.translate(-65, -37)
qp.drawText(p, '10')
qp.translate(65, 37)

qp.translate(-37, -65)
qp.drawText(p, '11')
qp.translate(37, 65)

qp.translate(0, -75)
qp.drawText(p, '12')
qp.translate(0, 75)
qp.end()
```

Római számok esetén pontosan ugyanezt csináljuk, ám azzal bonyolítva, hogy minden egyes szám beillesztése előtt forgatnunk kell, majd utána visszaforgatni.

Szintén, az óra számlapján a négyhez **IIII** lesz az elvárható **IV** helyett. Ez a szemközti oldalon lévő „**VIII**”, vagy nyolccal való szimmetria érdekében van így.

```
if numerals == 'roman':
font = QFont()
font.setFamily("Times New Roman")
font.setPointSize(14)
p = QPoint(-6, 6)

qp.translate(37, -65)
qp.rotate(30)
qp.drawText(p, 'I')
qp.rotate(-30)
qp.translate(-37, 65)

qp.translate(65, -37)
qp.rotate(60)
qp.drawText(p, 'II')
qp.rotate(-60)
qp.translate(-65, 37)

qp.translate(75, 0)
qp.rotate(90)
qp.drawText(p, 'III')
qp.rotate(-90)
qp.translate(-75, 0)

qp.translate(65, 37)
qp.rotate(-60)
qp.drawText(p, 'IIII')
qp.rotate(60)
qp.translate(-65, -37)

qp.translate(37, 65)
qp.rotate(-210)
qp.drawText(p, 'V')
qp.rotate(210)
qp.translate(-37, -65)

qp.translate(0, 75)
qp.rotate(-180)
qp.drawText(p, 'VI')
qp.rotate(180)
qp.translate(0, -75)

qp.translate(-37, 65)
qp.rotate(-150)
qp.drawText(p, 'VII')
qp.rotate(150)
qp.translate(37, -65)

qp.translate(-65, 37)
qp.rotate(-120)
qp.drawText(p, 'VIII')
qp.rotate(120)
qp.translate(65, -37)

qp.translate(-75, 0)
qp.rotate(-90)
qp.drawText(p, 'IX')
qp.rotate(90)
qp.translate(75, 0)

qp.translate(-65, -37)
qp.rotate(-60)
qp.drawText(p, 'X')
qp.rotate(60)
qp.translate(65, 37)

qp.translate(-37, -65)
qp.rotate(-30)
qp.drawText(p, 'XI')
qp.rotate(30)
qp.translate(37, 65)

qp.translate(0, -75)
qp.rotate(0)
qp.drawText(p, 'XII')
qp.translate(0, 75)
qp.end()
```

Igen elég hosszú, de szerintem nem fogod túl bonyolultnak találni.

A Style-hoz még változókat is hozzá kell adnunk.

Számok

E mögé

movement = 'sweep'

Írd be

numerals = 'arabic'

Az alternatív stílus eléréséhez ezt kell 'Roman'-ra változtatni.

Ezzel végeztünk a beállításokkal, ám a megváltoztatásukhoz a kódot kell szerkesztenünk. A Python számos módot ad az opciók futás közbeni fogadásához, ám ezek közül a legszebbet az argparse modul teszi lehetővé. Ez az, ami az elején bemutatott képernyőképek közül a részletes súgót hozza létre.

Az argparse modul használatához importálnunk kell azt, ezért a kódfájl elején lévő importáló részhez a következőt add hozzá:

import argparse

Ez az egyszerű sor elérést ad a modul összes nyilvános eljárásához és attribútumához.

Az 'if __name__' == '__main__' kódunk így néz ki most:

```
if __name__ == "__main__":
transparency = 'transparent'
marks = 'batons'
movement = 'sweep'
numerals = ‘arabic’
legend = 'Qt5 Clock'
xpos = 20
ypos = 20
theme = 'light'
app = QApplication(sys.argv)
clock = AnalogueClock()
clock.show()
if transparency == 'transparent':
clock.move(xpos, ypos)
else:
clock.move(xpos, ypos + 20)

sys.exit(app.exec_())
```

Ezt jelentősen módosítani kell. A helyettesítő kód elég hosszú, ezért Geany-ben a szokásos módon összecsukott formában fogom bemutatni. Azután az egyes szakaszokat bemutatom és elmagyarázom.

Először veszünk egy ArgumentParser-t az importált argparse modulból és elnevezzük parser-nek.

```
parser = argparse.ArgumentParser(
prog='qt5_aclock.py',
formatter_class=argparse.ArgumentDefaultsHelpFormatter,
description='Provides an analog clock with sweep or '
'stepping second hand, choice of numerals. '
'Dots or batons for minute marks, '
```

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser(
    parser.add_argument("-l",
    parser.add_argument("-n",
    parser.add_argument("-r",
    parser.add_argument("-d",
    parser.add_argument("-b",
    parser.add_argument("-s",
    parser.add_argument("-t",
    parser.add_argument("-x", dest='x',
    parser.add_argument("-y", dest='y',
    args = parser.parse_args()
    movement = 'step'          # the default
    numerals = ''              # no numbers
    theme = 'light'            # default theme
    marks = 'dots'             # dts not batons
    transparency = 'transparency'  # transparency on
    xpos, ypos = 0, 0          # starting position
    legend = args.l[:12]
    if args.s:
        movement = 'sweep'
    else:
        movement = 'step'
    if args.n:
        numerals = 'arabic'
    if args.r:
        numerals = 'roman'
    if args.d:
        theme = 'dark'
    if args.b:
        marks = 'batons'
    if args.t:
        transparency = 'transparent'
    if args.x:
        xpos = args.x
    if args.y:
        ypos = args.y

    app = QApplication(sys.argv)
    clock = AnalogueClock()
    clock.show()
    if transparency:
        clock.move(xpos, ypos)
    else:
        clock.move(xpos, ypos + 20)  # allow for titlebar
    sys.exit(app.exec_())
```

```
'light or dark theme and optional text field.',
epilog='copyright Casualsoft 2018')
```

A parser-nek meg kell adnunk az alkalmazás nevét, a formázás típusát (mi itt az alapbeállítást használjuk), adnunk kell egy leírást és egy opcionális, epilógusként ismert záró sort. Ez állítja be számunkra a parser-t. Most hozzáadjuk az opciókat, amiket az alkalmazást módosítására lehet használni.

```
parser.add_argument("-l",
help="permits cusomization by adding text to the \
clockface (maximum 12 characters). \
quote if spaces included\n",
type=str,
default='Qt5 Clock')
```

Az első opció a jelmagyarázat, amit az -l hív meg. A súgó leírja az opció jelentését és használatát: az opció átad egy sztringet a type objektumnak, ami ha nincs meghatározva, az alap 'Qt5 Clock' lesz. Vedd észre a fordított törtjel használatát a sorok folytatásához a súgó sztringben. A fordított törtjel a sor utolsó karaktere kell legyen.

A következő öt szakasz további opciókat ír le nagyon hasonló módon.

```
parser.add_argument("-r",
       help="adds Roman numerals to the display",
       action="store_true")
parser.add_argument("-d",
       help="apply the dark theme",
       action="store_true")
parser.add_argument("-b",
       help="Use batons for minute marks, \
       default is dots",
       action="store_true")
parser.add_argument("-s",
       help="changes the default stepping second hand\
       to a sweeping motion",
       action="store_true")
       parser.add_argument("-t",
       help="enables transparency - compositing must\
       be enabled and active",
       action="store_true")
```

Itt meghatározzuk az opciót, leírást adunk és megmondjuk a parser-nek, jegyezze meg, hogy ezt az opciót adtuk át.

X-et és y-t, a kiindulópont koordinátáit a változók integer típusként kapják meg egy alapbeállítás szerinti értékkel, amennyiben nincs más meghatározva az alkalmazásindító parancsban.

```
parser.add_argument("-x", dest='x',
       help="x startup position",
       action="store",
       type=int,
       default="10")
parser.add_argument("-y", dest='y',
       help="y startup position",
       action="store",
       type=int,
       default="10")
```

A parser által éppen összegyűjtött információkat az args változó menti el és ezt követően mi beállítjuk a default (alap) feltételeket.

```
args = parser.parse_args()
       movement = 'step' # the default
       numerals = '' # no numbers
       theme = 'light' # default theme
       marks = 'dots' # dots not batons
       transparency = 'transparency' # transparency on
       xpos, ypos = 0, 0 # starting position
       legend = args.l[:12]
```

A jelmagyarázatot az args.l tárolja. A méretkorlát miatt a jelmagyarázatnak csak az első 12 ([0-11]) karakterét használja fel.

Ezután ellenőrizzük az arg.s-ben szereplő értéket, hogy meghatározzuk a mozgás irányát a nagymutató számára.

```
if args.s:
movement = 'sweep'
Else:
movement = 'step'
```

Ezután, hasonló módon ellenőrizzük a több jellemzőt is.

```
if args.n:
numerals = 'arabic'
if args.r:
numerals = 'roman'
if args.d:
theme = 'dark'
if args.b:
marks = 'batons'
if args.t:
transparency = 'transparent'
if args.x:
xpos = args.x
```

```
if args.y:
ypos = args.y
```

Az argumentum elemző (parser) elvégezte dolgát, a szükséges jellemzők beállítva így a szokásos alkalmazásindító kóddal folytatjuk.

```
    app = QApplication(sys.argv)
    clock = AnalogueClock()

    # Single instance check
    Try:
        import socket
        s = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
        s.bind('\0postconnect_gateway_notify_lock_aclock')
    except socket.error as e:
        error_code = e.args[0]
        error_string = e.args[1]
        print("Error {}, {}. Exiting".format(error_code,
              error_string))
        QMessageBox.warning(clock,
        "Qt Analogue clock",
        "Process already running, Exiting!")
        sys.exit(0)

    clock.show()
    if transparency:
        clock.move(xpos, ypos)
    Else:
        clock.move(xpos, ypos + 20) # allow for titlebar

# subprocess.Popen('wmctrl -r "Qt5 Clock"
# -b add,skip_taskbar', shell=True)

    sys.exit(app.exec_())
```

Az egypéldányos kód az alap kód, amit az ébresztőhöz adtam – időzítő alkalmazás. Csak dobd oda be, hacsak nincs szükséged egynél több órára (más időzónában lévő barátok? A jelmagyarázatot használd az egyes órák azonosításához.)

Ha az óra unalmasan néz ki az indítópulton, átállítható a két legvégén álló sor megjegyzésből történő kiemelésével. Erre az openbox-os rendszeremben van szükségem. A működéséhez telepített wmctrl kell.

Ezzel egy, szerintem elég profi alkalmazást kaptunk. Amatőröktől nem is rossz.

Halmazok

A halmazok a Python alapvető elemeinek egyike, amivel még nem foglalkoztam. Ha tanultál matematikát, a halmazelmélet ismerős lehet, de ne hagyd, hogy ez félrevezessen. A halmazok Python-beli használata elég kidolgozott, végül is a Pythont kifejlesztő Guido van Rossum matematikus volt. Pythonban a halmazok felhasználása során leginkább azt a tulajdonságot használják ki, hogy egy halmaz elemei egyediek és ezért felhasználhatóak egy adathalmazban lévő ismétlődő elemek eltávolítására.

A halmazok módosíthatóak. Lezárt halmaz létrehozható oly módon, hogy az objektumok gyűjteményét átadjuk a frozenset() függvénynek. A frozenset-ek nem módosíthatóak. Már használtam halmazt, de nem gyakran. A személyes megjegyzéseim a halmazokkal kapcsolatban a következőek:

Halmazok és fix halmazok (frozenset)

Egy halmaz módosítható, ám a frozenset nem.

Egy halmazhoz csak hasítható objektumok adhatók. A hasítható objektum az, ami rendelkezik egy fix, az objektum léte során nem változó hasító értékkel.

A Python összes változtathatatlan objektuma hasítható, miközben a nem változtatható tárolók (mint a listák, vagy szótárak) nem.

A halmaz egyedi értékek rendezetlen tárolója. Halmazok csoportjára alkalmazható az egyesítés, közös rész, különbség és szimmetrikus különbség. A halmaz elemeit kapcsos zárójelpár {} tartalmazza és egymástól vessző választja el. Az egyelemű halmazokban lehet záró vessző, de nem kötelező.

Issubset (other), vagy set <= other	Vizsgálja, hogy a halmaz elemeit az other tartalmazza-e
set < other	Vizsgálja, hogy a set valódi részhalmaza-e az other-nek, vagyis set<= other és set! = other
issuperset (other), vagy set >= other	Vizsgálja, hogy a set tartalmazza-e az other összes elemét,
set > other	Vizsgálja, hogy az other valódi részhalmaza-e a set-nek, vagyis set >= other és set != other

Halmazok létrehozása

a_set = {1, 2, 3}
b_set = {3, 4, 5}

Halmaz készíthető listából, vagy leíróból.

a_set = set(list)

üres halmaz létrehozható ezzel

a_set = set()

A set függvénnyel ne kapcsos zárójelet használj, a {} üres könyvtárat hoz létre. A halmazok módosíthatóak, de a benne foglalt értékek nem változtathatóak meg.

Érték hozzáadása

a set.add még nem létező elemet ad hozzá, az elem értéke egyedi kell legyen.

Ser.add (a_set, 4) **==> {1, 2, 3, 4}**

A kísérlet létező érték halmazhoz adására nem csinál semmit.

Halmaz frissítése

set.update (a_Set, b_set) **==> {1, 2, 3, 4}**

Az egyik halmazban még nem létező elemeket hozzáadja a másikból.

Érték eltávolítása

set.discard eltávolít egy értéket. Nem létező elem nem csinál semmit.

set.remove eltávolít egy elemet. Nem létező elem hibajelzést ad.

Set.discard (a_set, 2) **==> (1, 3, 4, 5)**

set.pop letávolít egy értéket

set.pop (a_set) **==> ???**

mivel a halmaz rendezetlen, az eltávolított érték véletlenszerű - ** rossz **.

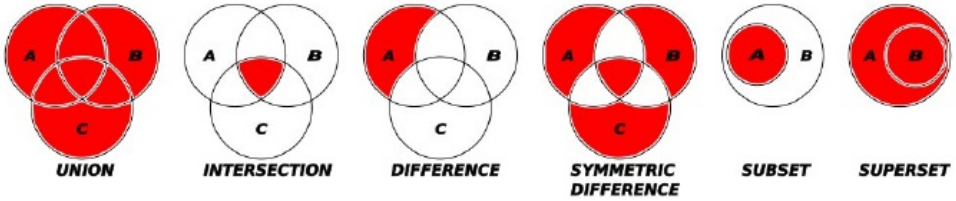
Ugyanakkor, üres halmaz popp-olása hibát eredményez.

Halmaz pucolása

A pucolás kiüríti a halmazt.

Set.clear (a_set) **==> set()**

Kombinatorikai halmazműveletek



a_set = (1, 2, 3, 4, 5)
b_set = (4, 5, 6, 7, 8)
c_set = (1, 5, 6, 7, 9)

Tagság ellenőrzése

2 in a_set **==> True (igaz)**
2 in b_set **==> False (hamis)**

A következő példákban mind az eljárás meghívása, mind a rövidebb operátor azonos eredményt ad vissza.

Halmaz union (egyesítés)

a_set .union (b_set) **==> (1, 2, 3, 4, 5, 6, 7, 8)**
a_set | b_set **==> (1, 2, 3, 4, 5, 6, 7, 8)**
a_set | b_set | c_set **==> (1, 2, 3, 4, 5, 6, 7, 8, 9)**

Halmaz intersection (közös rész)

A közös rész új halmazt ad azokkal az elemekkel, amiket mindkét, vagy összes halmaz tartalmaz.

a_set .intersection (b_set) ==> (4, 5)
a_set & b_set ==> (4, 5)
a_set & b_set & c_set ==> (5)

Halmaz difference (különbség)

A különbség új halmazt ad az összes olyan elemmel ami a-ban megvan, de b-ben (vagy c-ben) nincs.

a_set .difference (b_set) ==> (1, 2, 3)
a_set - b_set ==> (1, 2, 3)
a_set - b_set - c_set ==> (2, 3,)

Halmaz symmetric difference (szimmetrikus különbség)

A szimmetrikus különbség új halmazt ad az összes olyan elemmel, ami egy adott halmazban megvan.

a_set .symmetric_difference (b_set) ==> (1, 2, 3, 6, 7, 8)
a_set ^ b_set ==> (1, 2, 3, 6, 7, 8)
a_set ^ b_set ^ c_set ==> (2, 3, 5, 8, 9)

Boolean halmaz eljárások

A Boolean eljárások is támogatottak.

c_set = (5. 6. 7)
c_set.issubset (a_set) ==> False
c_set.issubset (b_set) ==> True
b_set.issubsuperset (c_set) ==> True

Üres halmaz az Boolean False, ugyanakkor minden más halmaz True.

További halmazműveletek

A következő műveletek támogatottak halmazokra.

A következő halmazokat használva:

```
set1 = {'England', 'Wales', 'Scotland', 'Ireland',
        'France', 'Germany','Italy', 'Sweden'}
set2 = {'Sweden', 'Norway', 'Finland'}
set3 = {'latvia', 'Estonia'}
set4 = {'Sweden', 'Norway'}
```

A halmaz elemeinek számának kiolvasása (sorszám).

len(set1) ==> 8

Tagság vizsgálata

```
'Spain' in set1              ==> True
'Spain' not in set1          ==> False
```

Tartalmak összehasonlítása

```
set1.isdisjoint(set2)        ==> False
set1.isdisjoint(set3)        ==> True
set4.issubset(set2)          ==> True
set2.issubset(set4)          ==> False
set2.issuperset(set4)        ==> True
set4 < set2                  ==> True
set4 > set2                  ==> False
set2 > set4                  ==> True
```

Kombinációk

```
set5 = set1.union(set4)
set5                         ==> {'Germany', 'France',
                                  'Scotland', 'Ireland', 'Wales',
                                  'Italy', 'Norway', 'Sweden',
                                  'England'}

set1.intersection(set2)      ==> {'Sweden'}
t2.difference(set1)          ==> {'Finland', 'Norway'}

set2.symmetric_difference(set1) ==> {'Germany', 'France',
                                     'Scotland', 'Finland',
                                     'Ireland', 'Wales', 'Italy',
                                     'Norway', 'England'}
```

-

A következők fix halmazokra (frozenset) nem érhetőek el.

```
set4.update(set3)
set4                        ==> {'Norway', 'Estonia', 'Sweden',
                                 'latvia'}

set1.intersection_update(set2)
set1                        ==> {'Sweden'}

set2.difference_update(set4)
set2                        ==> {'Finland'}

set4.symmetric_difference_update(set5)
set4                        ==> {'Germany', 'latvia', 'France',
                                 'Scotland', 'Wales', 'Ireland',
                                 'Italy', 'Estonia', 'England'}

set1                        ==> {'Sweden'}
set1.add('Lapland')
set1                        ==> {'Lapland', 'Sweden'}

set4.discard('Germany')
set4.discard('Spain')         # Not in set but no Error
set4.remove('Spain')        ==> Traceback (most recent call last):

File "", line 1, in <module>

KeyError: 'Spain'  # Exception raised

set4.remove('Scotland')
set4                        ==> {'latvia', 'France', 'Wales',
                                 'Ireland', 'Italy', 'Estonia',
                                 'England'}

set4.pop()                  ==> 'latvia' # Arbitrary element removed
set4.pop()                  ==> 'France'
set4.pop()                  ==> 'Wales'
set4                        ==> {'Ireland', 'Italy', 'Estonia',
                                 'England'}

set4.clear()
set4                        ==> set()
```

A szerkesztő megjegyzése: az Alkalmi Python cikksorozat kódjai letölthetőek innen.