Αντικειμενοστραφής Προγραμματισμός - Μέρος 2ο

Μιχάλης Δρακόπουλος

Ιούνιος 2020

Αντικείμενα ως ορίσματα συναρτήσεων

[1]:
```
def incr_int(n):
    n += 1
    print("n =", n)
```

[2]:
```
def main():
    x = 17
    incr_int(x)
    print('x =', x)
```

[3]:
```
class embedded_int:

    def __init__(self, n):
        self.value = n

    def set_value(self, n):
        self.value = n

    def get_value(self):
        return self.value

    def __str__(self):
        return str(self.value)
```

[4]:
```
def incr_embedded_int(N):
    N.set_value(N.get_value()+1)
    print("N =", N)
```

[5]:
```
def change_embedded_int(N):
    t = embedded_int(N.get_value() + 1)
    N = t
    print("N =", N)
```

[6]:
```
def main():
    y = embedded_int(17)
    incr_embedded_int(y)
    print("y =", y)
```

```
    change_embedded_int(y)
    print("y =", y)
```

Κλάση για ρητούς αριθμούς

[7]:
```
import math

class Rational:

    def __init__(self, x = 0, y = 1):
        self.num = x
        if y == 0:
            self.den = 1
        else:
            self.den = abs(y)
            if y < 0: self.num = -self.num
            self.simplify()

    def get_num(self): return self.num

    def get_den(self): return self.den

    def __str__(self):
        if self.den == 1:
            return str(self.num)
        else:
            return str(self.num) + '/' + str(self.den)

    def simplify(self):
        g = math.gcd(abs(self.num), abs(self.den))
        self.num //= g
        self.den //= g

    def add(self, other):
        return Rational(self.num*other.den + self.den*other.num, self.den*other.
↪den)

    def times(self, other):
        return Rational(self.num*other.num, self.den*other.den)

    def equals(self, other):
        return self.num == other.num and self.den == other.den
```

[8]:
```
x = Rational(10, 30)
y = Rational(5)
z = Rational(1, 2)
```

[9]:
```
a = x.add(y)
print(x, '+', y, '=', a)
```

```
1/3 + 5 = 16/3
```

[10]:
```
b = x.add(y).add(z)
print(x, '+', y, '+', z, '=', b)
```

```
1/3 + 5 + 1/2 = 35/6
```

[11]:
```
a = x.times(y)
print(x, '*', y, '=', a)
```

```
1/3 * 5 = 5/3
```

[12]:
```
d = x.add(y).times(z)
print(x, '+', y, '*', z, '=', d)
```

```
1/3 + 5 * 1/2 = 8/3
```

d = x + y * z

[13]:
```
import math

class Rational:

    def __init__(self, x = 0, y = 1):
        self.num = x
        if y == 0:
            self.den = 1
        else:
            self.den = abs(y)
            if y < 0: self.num = -self.num
            self.simplify()

    def get_num(self): return self.num

    def get_den(self): return self.den

    def __str__(self):
        if self.den == 1:
            return str(self.num)
        else:
            return str(self.num) + '/' + str(self.den)

    def simplify(self):
        g = math.gcd(abs(self.num), abs(self.den))
        self.num //= g
        self.den //= g
```

```
    def __add__(self, other):
        #return Rational(self.num*other.den + self.den*other.num, self.
↪den*other.den)
        return Rational(42, 1)

    def __mul__(self, other):
        return Rational(self.num*other.num, self.den*other.den)

    def __eq__(self, other):
        return self.num == other.num and self.den == other.den
```

[14]:
```
x = Rational(10, 30)
y = Rational(5)
z = Rational(1, 2)
```

[15]:
```
a = x + z
print(a)
```

```
42
```

[16]:
```
print(y*z*x)
```

```
5/6
```

[17]:
```
x == z
```

[17]:
```
False
```

[18]:
```
dir(int)
```

[18]:
```
['__abs__',
 '__add__',
 '__and__',
 '__bool__',
 '__ceil__',
 '__class__',
 '__delattr__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__float__',
 '__floor__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getnewargs__',
```

```
'__gt__',
'__hash__',
'__index__',
'__init__',
'__init_subclass__',
'__int__',
'__invert__',
'__le__',
'__lshift__',
'__lt__',
'__mod__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__or__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rlshift__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rrshift__',
'__rshift__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__trunc__',
'__xor__',
'as_integer_ratio',
'bit_length',
'conjugate',
```

```
'denominator',
'from_bytes',
'imag',
'numerator',
'real',
'to_bytes']
```