I Zagadnienia do opanowania

1. Co to są moduły i pakiety? Jak je wykorzystujemy w projekcie?
2. Obiekty i klasy:
   a) Jak definiujemy klasy?
   b) Jak tworzymy obiekty danej klasy?
   c) Co to są metody? Jak je definiujemy i jak je stosujemy na obiektach?
   d) Jak odwołujemy się do atrybutów (pól i metod) obiektu?
   e) Co to są metody specjalne (np. __init__, __add__, __str__ itp.)?
   f) Co to jest dziedziczenie i polimorfizm? Jak to działa?

II Rozgrzewka...

1. Moduły i pakiety:
   a) Sprawdź działanie poniższego skryptu?

```
import math
i=float(input('Podaj liczbę:'))
print('Pierwiastek liczby',i,'=',math.sqrt(i))
print('jej logarytm =', math.log(i))
print('Obwód kola o promieniu',i,'=',2*math.pi*i)
```

   c) Sprawdź jakie inne funkcje matematyczne są dostępne w module **`math`**. Jaka funkcja zwraca odległość punktu od początku układu wsp. w metryce euklidesowej?
   d) Sprawdź szybkość działania różnych funkcji obliczających silnię z liczby. Która z metod jest najszybsza, a która najwolniejsza (porównaj wyniki np. dla 5! i 100!)?

```
from math import factorial
from time import time
def silnia_rekur(n):
    if n==0 or n==1:
        return 1
    else:
        return n*silnia(n-1)
def silnia_iter(n):
    wynik=1
    for i in range(1,n+1):
        wynik*=i
    return wynik
i=int(input('Podaj liczbę całkowitą:'))
p_rekur = time()
w_rekur = silnia_rekur(i)
k_rekur = time()

p_iter = time()
w_iter = silnia_iter(i)
k_iter = time()

p_mat = time()
w_mat = factorial(i)
k_mat = time()

print(str(i)+"!, wynosi:", w_rekur)
print('Obliczenia rekurencyjnie
trwały:',k_rekur-p_rekur,'sekund')
print('Obliczenia iteracyjnie trwały:',k_iter-p_iter,'sekund')
print('Obliczenia math.factorial trwały:',k_mat-p_mat,'sekund')
```

2. Klasy:
   a) Poniżej przestawiono fragment klasy **wektor**, do czego służą poszczególne metody tej klasy?

```
class wektor(object):

    def __init__(self, x=1,y=1):
        self.x=x
        self.y=y

    def dlugosc(self):
        return math.hypot(self.x,self.y)

    def __add__(self,other):
        return wektor(self.x+other.x, self.y+other.y)

    def __str__(self):
        return ("Wektor(%g,%g)" % (self.x,self.y))

    def __repr__(self):
        return ("Wektor(%g,%g)" % (self.x,self.y))
```

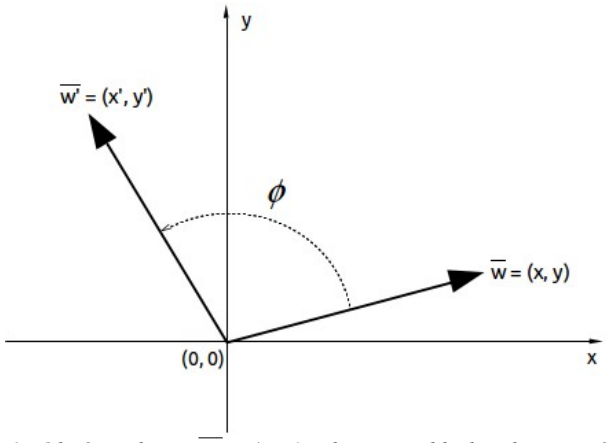
   b) Co będzie wynikiem działania następującego skryptu (zakładając, że powyższy fragment zawierający klasę **wektor** jest fragmentem tego skryptu)?

```
w1 = wektor()
w2 = wektor(2,2)
w3 = w1+w2
print('wektor w1=', w1, 'ma długość', w1.dlugosc())
print('wektor w2=', w2, 'ma długość', w2.dlugosc())
print('wektor w3=', w3, 'ma długość', w3.dlugosc())
```

   c) Wiedząc, że w układzie kartezjańskim obrót wektora $\overline{w} = (x, y)$, o kąt ϕ (patrz rys. 1) powoduje zmianę jego współrzędnych według następujących zależności:

$$x' = x \cdot \cos(\phi) - y \cdot \sin(\phi)$$
$$y' = x \cdot \sin(\phi) + y \cdot \cos(\phi)$$

   uzupełnij klasę **wektor** o metodę **obrot**, która będzie obracała wektor o zadany kąt (wyrażony w stopniach). Przetestuj działanie tej klasy!



Rys. 1: Obrót wektora $\overline{w} = (x, y)$ o kąt ϕ w układzie kartezjańskim

III Zadania do rozwiązania:

1. Utwórz nowy skrypt i zdefiniuj w nim klasę **Punkt**, która będzie definiowała obiekt (punkt) w przestrzeni dwuwymiarowej (domyślnie na początku układu współrzędnych (0,0)) oraz będzie posiadała dwie metody: **odleglosc** – zwracającą odległość punktu od początku układu współrzędnych oraz **dystans** – zwracająca odległość między dwoma punktami (obiektami należącymi do klasy **Punkt**). Ponadto zdefiniuj w klasie **Punkt** dwie metody specjalne **__repr__** oraz **__str__** , które pozwolą na wyświetlenie współrzędnych punktu, np.:

```
>>> p1 = Punkt(1.2, 2)
>>> p1
(1.2, 2)
>>> print(p1)
Punkt(1.2, 2)
```

   Uzupełnij skrypt (stwórz kilka punktów – tj. obiektów klasy **Punkt**) i przetestuj jego działanie.
2. Zdefiniuj w powyższym skrypcie klasę **Kolo**, która będzie dziedziczyła po klasie **Punkt** i będzie tworzyła w przestrzeni dwuwymiarowej (na płaszczyźnie) koło o środku w punkcie (x, y) i o promieniu r (domyślnie: środek koła w środku układu współrzędnych (0,0), promień jednostkowy r = 1).
3. Uzupełnij klasę **Kolo** o metody: **obwod** – zwracającą obwód koła oraz **pole** – zwracającą pole powierzchni koła oraz dwie metody specjalne: **__str__** i **__repr__** o działaniu podobnym jak w p. 2. Uzupełnij następnie skrypt tak, aby można było pokazać poprawną implementację klas i metod (zdefiniuj obiekty, przetestuj działanie metod na obiektach).
4. Uzupełnij klasę **Kolo** o metodę **przesun**, która będzie przesuwała kolo na płaszczyźnie o podany wektor (argument metody to krotka (x,y) lub lista [x,y] ze współrzędnymi wektora), np.:

```
>>> k1 = kolo(2,0,1)
>>> k1
(2,0,1)
>>> k1.przesun((1,2))
>>> k1
(3,2,1)
>>> k1.przesun(([-1,-1]))
>>> k1
(2,1,1)
```

5. Uzupełnij klasę **Kolo** o metodę **cz_wsp**, która będzie sprawdzała, czy dwa koła nie zachodzą na siebie (nie mają części wspólnych) i zwracała odpowiednio wartość **True** lub **False**.