Algorytmy

- Liczby doskonałe – to takie liczby gdzie suma ich dzielników jest równa tej liczbie:

```
def doskonale(a):
    suma = 0 # zmienna pomocnicza przechowujaca sume
    for i in range(1, a):
        if a%i==0: # w petli sprawdzamy dzielniki liczby i je sumujemy
            suma+=i
    if suma==a: #jezeli suma dzielnikow jest rowna wskazanej liczbie to jest ona doskonola
        return True
    else:
        return False
```

- Liczby pierwsze – to takie liczby które są podzielne tylko przez 1 i samą siebie:

```
def pierwsze(n):
    for i in range(2, n):
        if n%i==0:
            return False
    return True
```

- Największy wspólny dzielnik (Euklides)

```
def nwd(a, b):
    while a!=b:
        if a>b:
            a=a-b
        else:
            b=b-a
    return a
```

rekurencyjnie

```
def nwd_rekurencja(a, b):
    if b!=0:
        return nwd_rekurencja(b, a%b)
    return a
```

- Szybkie potęgowanie – podnoszenie liczby a do potęgi b

```
def potegowanie(liczba, potega):
    if potega == 0:
        return 1
    return potegowanie(liczba, potega-1) * liczba
```

- Silnia – silnia dla n liczby naturalnej

```
def silnia(n):
    if n == 0 or n == 1:
        return 1
    else:
        sil = 1
        for i in range(2, n+1, 1):
            sil *=i
        return sil
```

rekurencyjnie

```
### rekurencyjnie
def sil_rek(n):
    if n > 1:
        return n*sil_rek(n-1)
    elif n in (0,1):
        return 1
```

Fibonacci

```
def fibb_iter(ilosc):
    fibb = []
    a = 0
    b = 1
    for i in range(0, ilosc):
        fibb.append(b)
        b += a
        a = b - a
    return fibb
```

rekurencyjnie

```
def fibb_reku(n):
    if n < 3:
        return 1
    return fibb_reku(n - 2) + fibb_reku(n - 1)
```

Sito Erastotenesa

```
def sito(tab):
    n = len(tab)
    i = 0
    while n>0:
        #print("iteracja "+str(i+1))
        x = tab[i]
        buff = x
        for z in range(n):
            for y in tab[i+1::]:
                #print("Liczba "+str(y)+" dzielnik "+str(x))
                if y%x==0:
                    tab.remove(y)
                    n-=1
            x+=buff
        if n>0:
            i+=1
            n-=1
```

Quick sort

```
def quick_sort(tab):
    if len(tab) < 2:
        return tab
    else:
        pivot = tab[0]
        less = [x for x in tab[1:] if x < pivot]
        greater = [x for x in tab[1:] if x > pivot]
        return quick_sort(less) + [pivot] + quick_sort(greater)
```

Szukanie binarne

```
def binary_search(tab, x):
    low = 0
    high = len(tab)-1
    while(low<=high):
        mid = (low+high) #srodek
        element = tab[mid] # dodatowa zmienna element to zmienna buforowa przy
jmujacy ze zbioru naszego srodkowa wartosc
        if element==x:
            return element
        if element>x:
            high=mid-1
        else:
            low = mid+1
    return None
```

Szyfr cezara

```
def szyfr_cezara(klucz, slowo):
    buff = ""
    klucz = klucz % 26
    for x in slowo:
        if x == ' ':
            buff+= ' '
        elif ord(x)+klucz>90:
            buff+=chr(ord(x)+klucz-26)
        else:
            buff+=chr(ord(x)+klucz)
    return buff
```

Palindrom

```
def palindrom(text):
    n = len(text)
    p = 0
    for i in range(0, int(n/2), 1):
        if(text[i] == text[n-i-1]):
            p+=1
    if(p==int(n/2)):
        print("It's palindrom!")
    else:
        print("It's not palindrom :(")
```

Anagram

```
def anagram(text):
    n = len(text)
    for i in range(n-1, -1, -1):
        print(text[i])
```

Selection sort

```
### Zasada brzmi: szukanie elementu minimalnego
def selection_sort(tab):
    dl = len(tab)-1
    x=tab[0]
    for i in range(dl-1):
        if tab[i]<x:
            x = tab[i]
        for j in range(i+1, dl):
            tab[i] = x
```