# Υπολογιστική Γεωμετρία
# Πρώτη Εργασία

Σιώρος Βασίλειος - 1115201500144
Ανδρινοπούλου Χριστίνα - 1115201500006

Μάρτιος 2020

**1. Implement an algorithm that takes as input three points in the plane. checks that they form a triangle and whether the interior of the triangle contains the origin (0, 0) or not.**

**2. Given a circle of radius r in the plane with (0, 0) as center, implement an algorithm that finds the total lattice points on the circumference. Lattice Points are points with integer coordinates.**

### Thought Process

Σε ένα καρτεσιανό σύστημα συντεταγμένων, ένας κύκλος με κέντρο το σημείο (a, b) και ακτίνα r είναι ένα σχήμα το οποίο αποτελείται από όλα τα σημεία των οποίων οι συντεταγμένες ικανοποιούν την εξίσωση

$$(x-a)^2 + (y-b)^2 = r^2$$

Από την παρπάνω εξίσωση, μπορούμε να συμπεράνουμε, ότι οποιοδήποτε σημείο, του οποίου η Ευκλείδια απόσταση από το κέντρο του κύκλου είναι μεγαλύτερη της ακτίνας του, βρίσκεται εκτός του κύκλου.

Αν ο κύκλος έχει ως κέντρο την αρχή των αξόνων, δηλαδή το σημείο $(0, 0)$, τότε η παραπάνω εξίσωση παίρνει την εξής απλούστερη μορφή

$$x^2 + y^2 = r^2$$

Σε αυτή την περίπτωση, είτε η τετμημένη είτε η τεταγμένη ενός σημείου αρκεί να είναι μεγαλύτερη της ακτίνας του έτσι ώστε να μην ανήκει στον κύκλο.

Ως εκ τούτου, τα σημεία πλέγματος ενός κύκλου ακτίνας r με κέντρο το (0, 0) είναι υποσύνολο του συνόλου που απαρτίζεται από σημεία με ακέραιες συνταγμένες στο εύρος $[-r, +r]$ και στις δύο διαστάσεις, με την πρόσθετη συνθήκη να ικανοποιούν την παραπάνω εξίσωση έτσι ώστε να βρίσκονται επί της περιφέρειάς του.

Για παράδειγμα, για έναν κύκλο ακτίνας 1 με κέντρο το (0, 0) αρκεί να ελέγξουμε ποιά από τα σημεία εντός του συνόλου

$$\{(-1, 0), (0, -1), (0, +1), (+1, 0)\}$$

ικανοποιούν την εξίσωση

$$x^2 + y^2 = 1$$

### Implementation

Δεδομένου ότι δεν ήταν ξεκάθαρο, αν το πρόβλημα ζητούσε μόνο το πλήθος ή και τα ακριβή σημεία πλέγματος, αποφασίσαμε να ακολουθήσαμε την παρακάτω προσέγγιση η οποία μπορεί να χρησιμοποιηθεί για την εύρεση και των δύο όπως θα δείτε στην συνέχεια.

```python
def lattice_points(radius):
    points = []

    for x in range(-radius, radius + 1):
        for y in range(-radius, radius + 1):
            if x ** 2 + y ** 2 == radius ** 2:
                points.append((x, y))

    return points
```

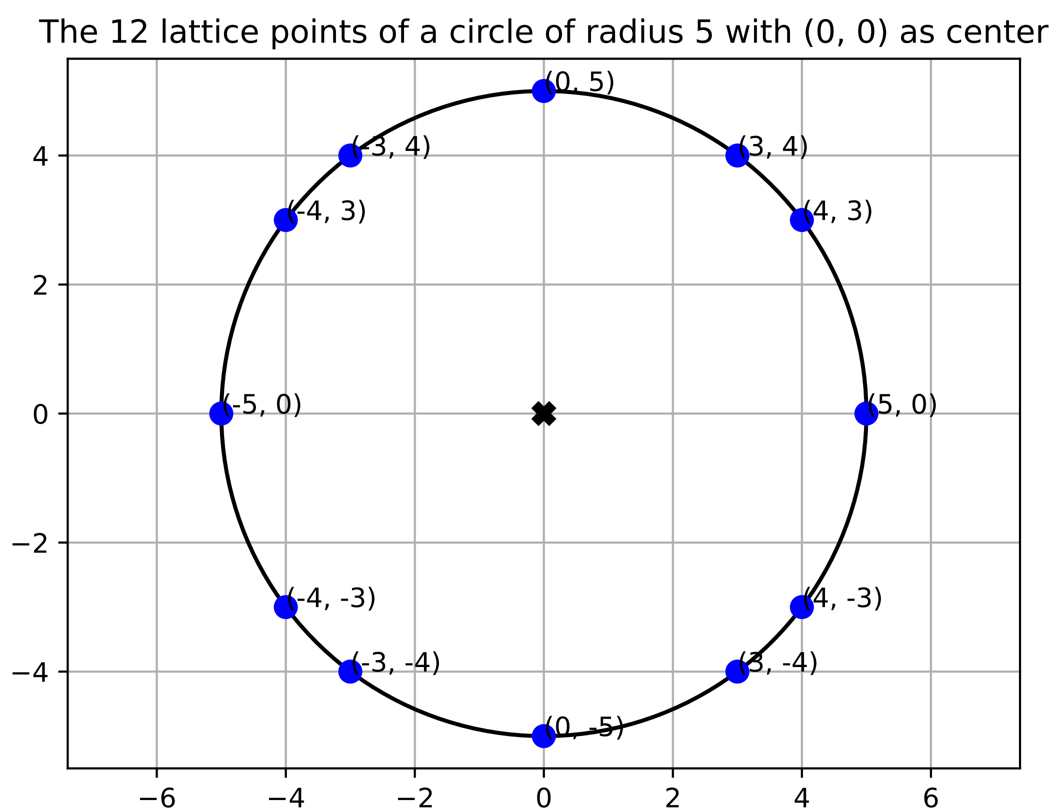
**Running the code**

```
$ python -m virtualenv .env
$ source .env/Scripts/activate
$ pip install -r requirements.txt
$ python exercise2.py -h
usage: exercise2.py [-h] -r RADIUS [-s SAMPLE]

Lattice Points Visualizer

optional arguments:
  -h, --help            show this help message and exit
  -r RADIUS, --radius RADIUS
                        The radius of the circle
  -s SAMPLE, --sample SAMPLE
                        The number of points to be used when generating the
                        circle
$ python exercise2.py -r 5
```

**Example Usage**



**3. Implement the incremental 2D algorithm for computing the convex hull of a finite set of points in the plane.**

## 4. Implement the gift wrap algorithm for computing the convex hull of a finite set of points in the plane .

### Thought Process

Ο **Gift Wrapping** αλγόριθμος, που στην περίπτωση των δύο διαστάσεων είναι επίσης γνωστός και ως ο **Jarvis March** αλγόριθμος, μπορεί να περιγραφεί από τα εξής απλά βήματα

1. Δεδομένου ενός συνόλου σημείων στον $\mathbb{R}^2$, επιλέγει ένα σημείο το οποίο είναι γνωστό ότι ανήκει στο **Convex Hull** του συνόλου, όπως π.χ. το αριστερότερο σημείο μεταξύ των δεδομένων σημείων.

2. Στη συνέχεια, επιλέγει ως επόμενο σημείο το σημείο για το οποίο, όλα τα υπόλοιπα σημεία βρίσκονται δεξιά της ευθείας που ορίζουν το τρέχον και το σημείο αυτό.

3. Αν το τρέχον σημείο ταυτίζεται με το αρχικά επιλεχθέν σημείο, τότε τερμάτησε καθώς το **Convex Hull** υπολογίστηκε και αποτελείται από το σύνολο επιλεχθέντων σημείων έως τώρα. Διαφορετικά επανάλαβε το βήμα 2.

**Implementation**

Ορίσαμε 3 βοηθητικές συναρτήσεις

- **orientation**: Υπολογίζει τον προσανατολισμο των δύο διανυσμάτων που ορίζουν τα 3 σημεία στην είσοδό της
- **counterclockwise**: Με τη βοήθεια της συνάρτηση **orientation** επιστρέφει αν τα τρία αυτά σημεία είναι τοποθετημένα με την αντίστροφη φορά του ρολογιού.
- **between**: Με τη βοήθεια της συνάρτηση **orientation** επιστρέφει αν τα τρία αυτά σημεία είναι συνευθειακά.

```python
def orientation(p1, p2, p3):
    return (p3[1] - p1[1]) * (p2[0] - p1[0]) - (p2[1] - p1[1]) * (p3[0] - p1[0])


def counterclockwise(p1, p2, p3):
    return orientation(p1, p2, p3) > 0


def between(p1, p2, p3):
    if orientation(p1, p2, p3) != 0:
        return False

    if min(p1[0], p3[0]) > p2[0] or p2[0] > max(p1[0], p3[0]):
        return False

    if min(p1[1], p3[1]) > p2[1] or p2[1] > max(p1[1], p3[1]):
        return False

    return True
```

Τέλος η συνάρτηση **jarvis** είναι υπεύθυνη για τον υπολογισμό του **Convex Hull**, του συνόλου σημείων που δέχεται στην είσοδό της και αποτελεί μία μικρή επέκταση του παραπάνω αλγορίθμου, έτσι ώστε να λαμβάνει υπ' όψιν και την ακραία περίπτωση 3 συνευθειακών σημείων.

```python
def jarvis(points):
    if len(points) < 3:
        return []

    points = sorted(points)

    convex_hull = [points[0]]

    current = None
    while True:
        current = points[0]
        for point in points[1:]:
            if current == convex_hull[-1] or \
                between(convex_hull[-1], current, point) or \
                    counterclockwise(convex_hull[-1], current, point):
                current = point

        if current == convex_hull[0]:
            break

        convex_hull.append(current)

    return convex_hull
```

**Running the code**

```
$ python -m virtualenv .env
$ source .env/Scripts/activate
$ pip install -r requirements.txt
$ python exercise4.py -h
usage: exercise4.py [-h] -n NUMBER [-x XRANGE [XRANGE ...]]
                    [-y YRANGE [YRANGE ...]]

Visualizing the Convex Hull of a given set of points with the help of the
Jarvis March Algorithm

optional arguments:
  -h, --help            show this help message and exit
  -n NUMBER, --number NUMBER
                        The number of points to be randomly generated
  -x XRANGE [XRANGE ...], --xrange XRANGE [XRANGE ...]
                        The horizontal axis' range of values
  -y YRANGE [YRANGE ...], --yrange YRANGE [YRANGE ...]
                        The vertical axis' range of values
$ python exercise4.py -n 20
```

**Example Usage**