ΤΕΙ Δυτικής Ελλάδας
Τμήμα μηχανικών πληροφορικής Τ.Ε

# ΕΠΕΞΗΓΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΜΕ ANIMATION ΣΕ PYTHON ΓΙΑ ΕΚΠΑΙΔΕΥΤΙΚΟ ΣΚΟΠΟ

Πτυχιακή εργασία του
Αλίαϊ Γιουλιάν-Νίκος

Εισηγητής καθηγητής
Δρ. Τσακανίκας Βασίλης

Φεβρουάριος 2016

o

**Ευχαριστίες**

Ξεκινώντας θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου κύριο Βασίλη Τσακανίκα για την εμπιστοσύνη που μου έδειξε επιλέγοντας με για την υλοποίηση τις παρούσας εργασίας, δίνοντας μου έτσι την δυνατότητα να ασχοληθώ με ένα τόσο ενδιαφέρων θέμα.
Θα ήθελα επίσης να τον ευχαριστήσω για τη βοήθεια, την καθοδήγηση και τις χρήσιμες συμβουλές που μου παρείχε..

Ένα μεγάλο ευχαριστώ οφείλω να πω και στην οικογένεια μου που με στήριξε όλον αυτών τον καιρό και ήταν συνέχεια δίπλα μου με κάθε τρόπο δίνοντας μου κουράγιο να συνεχίσω, σας ευχαριστώ όλους.

# Μαθαίνοντας Python

## 1 Βημα πρώτο, HTML

Η HTML (αρχικοποίηση του αγγλικού HyperText Markup Language, ελλ. Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων.

Στη παρούσα διπλωματική αρχικά χρησιμοποιήσαμε και βασικά στοιχεία της HTML αλλά και στοιχεία από CSS για την διαμόρφωση του σάιτ μας.

### 1.1 Τι σημαίνει CSS;

Η CSS *(Cascading Style Sheets-Διαδοχικά Φύλλα Στυλ)* ή ( αλληλουχία φύλλων στυλ ) είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

### 1.2 Bootstrap

Το Bootstrap είναι μια συλλογή εργαλείων ανοιχτού κώδικα (Ελεύθερο λογισμικό) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλων στοιχείων του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις JavaScript. Είναι το πιο δημοφιλές πρόγραμμα στο GitHub [1] και έχει χρησιμοποιηθεί από τη NASA και το MSNBC, μεταξύ άλλων.

### 1.3 Το site μας

Αρχικά ορίσαμε τα μεταδεδομένα μας και τις πηγές των css ορισμάτων

```
<head>

  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="">
  <meta name="author" content="">
```

```
  <title>Εκμάθηση Python </title>

  <!-- Bootstrap Core CSS -->
  <link href="css/bootstrap.min.css" rel="stylesheet">

  <!-- Custom CSS -->
  <link href="css/clean-blog.min.css" rel="stylesheet">

  <!-- Custom Fonts -->
  <link href="http://maxcdn.bootstrapcdn.com/font-awesome/4.1.0/css/font-awesome.min.css"
rel="stylesheet" type="text/css">
  <link href='http://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic' rel='stylesheet'
type='text/css'>
  <link
href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700italic,800ital
ic,400,300,600,700,800' rel='stylesheet' type='text/css'>



</head>
```

Έπειτα σχεδιάσαμε το βασικό σχέδιο πλοήγησης στο σαιτ μας

```
<!-- Navigation -->
  <nav class="navbar navbar-default navbar-custom navbar-fixed-top">
    <div class="container-fluid">
      <!-- Brand and toggle get grouped for better mobile display -->
      <div class="navbar-header page-scroll">
        <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-
example-navbar-collapse-1">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <a class="navbar-brand" href="index.html">Το σαιτ μου</a>
      </div>

      <!-- Collect the nav links, forms, and other content for toggling -->
      <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
        <ul class="nav navbar-nav navbar-right">
          <li>
            <a href="index.html">Αρχικη</a>
          </li>
          <li>
            <a href="#">Πληροφοριες</a>
```

```
            </li>
            <li>
              <a href="post.html">Αλγοριθμος του Καισαρα</a>
            </li>

          </ul>
        </div>
        <!-- /.navbar-collapse -->
      </div>
      <!-- /.container -->
    </nav>
```

Και στη συνέχεια τη βασική δομή του σαιτ η οποια είναι κοινή και για όλες τις σελίδες του σαιτ

```
<header class="intro-header" style="background-image: url('img/home-bg.jpg')">
      <div class="container">
        <div class="row">
          <div class="col-lg-8 col-lg-offset-2 col-md-10 col-md-offset-1">
            <div class="site-heading">
              <h1>Εκμάθηση Python</h1>
              <hr class="small">
              <span class="subheading">Μια προσπάθεια εκμάθησης της Python μέσα από
διαδραστικά παραδείγματα</span>
            </div>
          </div>
        </div>
      </div>
    </header>

    <!-- Main Content -->
    <div class="container">
      <div class="row">
        <div class="col-lg-8 col-lg-offset-2 col-md-10 col-md-offset-1">
          <div class="post-preview">
            <a href="post.html">
              <h2 class="post-title">
                Ο Αλγόριθμος του Κάισαρα
              </h2>
              <h3 class="post-subtitle">
                Ένας αρχαίος αλγόριθμος κρυπτογράφησης
              </h3>
            </a>
            <p class="post-meta">Δημιουργήθηκε από τον <a href="#">Νίκο</a> τον Φεβρουάριο
του 2016</p>
          </div>
          <hr>
          <div class="post-preview">
```

```
            <a href="post.html">
              <h2 class="post-title">
                Ο Αλγόριθμος Quicksort
              </h2>
            </a>

          </div>
          <hr>

          <!-- Pager -->
          <ul class="pager">
            <li class="next">
              <a href="https://gist.github.com/NikosAliai">Οι κώδικες μου στο GitHub&rarr;</a>
            </li>
          </ul>
        </div>
      </div>
    </div>

    <hr>

    <!-- Footer -->
    <footer>
      <div class="container">
        <div class="row">
          <div class="col-lg-8 col-lg-offset-2 col-md-10 col-md-offset-1">
            <ul class="list-inline text-center">
              <li>
                <a href="#">
                  <span class="fa-stack fa-lg">
                    <i class="fa fa-circle fa-stack-2x"></i>
                    <i class="fa fa-twitter fa-stack-1x fa-inverse"></i>
                  </span>
                </a>
              </li>
              <li>
                <a href="#">
                  <span class="fa-stack fa-lg">
                    <i class="fa fa-circle fa-stack-2x"></i>
                    <i class="fa fa-facebook fa-stack-1x fa-inverse"></i>
                  </span>
                </a>
              </li>
              <li>
                <a href="#">
                  <span class="fa-stack fa-lg">
                    <i class="fa fa-circle fa-stack-2x"></i>
                    <i class="fa fa-github fa-stack-1x fa-inverse"></i>
```

```
                    </span>
                  </a>
                </li>
              </ul>
              <p class="copyright text-muted">Copyright &amp;copy; Python  Website 2016</p>
            </div>
          </div>
        </div>
      </footer>
```

## 1.4 Ο αλγόριθμος του Καίσαρα

Στη σελίδα αυτή χρησιμοποιήσαμε javascript για να δημιουργήσουμε το συγκεκριμένο animation

```
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideToggle("slow");
  });
});
</script>
<script>
$(document).ready(function(){
  $("#flip1").click(function(){
    $("#panel1").slideToggle("slow");
  });
});
</script><script>
$(document).ready(function(){
  $("#flip2").click(function(){
    $("#panel2").slideToggle("slow");
  });
});
</script>
```

Και

```
<script type="text/javascript">
startRotation = 0;
wheelIsRotating = false;

function getAngle(e) {
  var wheeloffset = $('#wheelimg').offset();
  var wheelwidth = $('#wheelimg').width();
  var wheelheight = $('#wheelimg').height();
  var originx = wheeloffset.left + (wheelwidth / 2);
  var originy = wheeloffset.top + (wheelheight / 2);

  var x = e.pageX - originx;
```

```
    var y = e.pageY - originy;

    if (x == 0) {
        if (y <= 0) {
            return 90.0;
        }
        else {
            return 270.0;
        }
    }
    var slope = (y / x);
    var angle = Math.atan( slope ) * (180 / 3.141592);

    if (y >= 0 && x >= 0) {
        angle = (90 - angle) + 270.0;
    }
    if (y >= 0 && x < 0) {
        angle = -angle + 180.0;
    }
    if( y < 0 && x < 0) {
        angle = (90 - angle) + 90.0;
    }
    if (y < 0 && x >= 0) {
        angle = -angle;
    }

    if (angle == 360.0) {
        return 0.0;
    }
    else {
        return angle;
    }
}


function clickCipherWheel(e) {
    var angle = getAngle(e);

    if (wheelIsRotating) {
        wheelIsRotating = false;
        $('#wheelinfo').html('Click wheel to rotate.');
        adjustment = parseInt((startAngle - angle) / 6.9230);
        startRotation = mod(adjustment + startRotation, 52)
        return;
    }
    else {
        startAngle = angle;
        wheelIsRotating = true;
```

```
      $('#wheelinfo').html('Click wheel to stop rotating.');
   }
}

function mod(a, b) {
   return ((a % b) + b) % b;
}

function showRotation(n) {
   $('#wheelimg').css('background-position', (n * -400).toString() + 'px 0px');

   // # rotate the number line at the bottom:
   var notLinedUp = (n % 2 == 1);
   var charCode = 65 + Math.floor((52 - n) / 2);
   var i = 0;
   var s = '';

   if (!notLinedUp) {
      s += ' ';
   }

   while (i < 26) {
      if ((charCode - 65) > 25) {
         charCode = ((charCode - 65) % 26) + 65;
      }

      schar = String.fromCharCode(charCode);

      if (charCode == 65) {
         s += ' <u>' + schar + '</u> ';
      } else {
         if (schar.length == 1) {
            schar = ' ' + schar;
         }
         s += schar + ' ';
      }
      i += 1;
      charCode += 1;

   }
   $('#shiftedLetters').html(s);
}

function rotateCipherWheel(e) {
   if (!wheelIsRotating) {
      return;
   }
   adjustment = parseInt((startAngle - getAngle(e)) / 6.9230);
```

```
   showRotation(mod(adjustment + startRotation, 52));
}

$('#wheelimg').mousemove( rotateCipherWheel );
$('#wheelimg').click( clickCipherWheel );
showRotation(0);
</script>
```

# 2 Τι είναι η python;

Η **Python** είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού η οποία δημιουργήθηκε από τον Ολλανδό Γκβίντο βαν Ρόσσουμ (Guido van Rossum) το 1990. Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της και το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα απ'ότι θα ήταν δυνατόν σε γλώσσες όπως η C++ ή η Java. Διακρίνεται λόγω του ότι έχει πολλές βιβλιοθήκες που διευκολύνουν ιδιαίτερα αρκετές συνηθισμένες εργασίες και για την ταχύτητα εκμάθησης της.

Οι διερμηνευτές της Python είναι διαθέσιμοι για εγκατάσταση σε πολλά λειτουργικά συστήματα, επιτρέποντας στην Python την εκτέλεση κώδικα σε ευρεία γκάμα συστημάτων. Χρησιμοποιώντας εργαλεία τρίτων, όπως το Py2exe ή το Pyinstaller, ο κώδικας της Python μπορεί να πακεταριστεί σε αυτόνομα εκτελέσιμα προγράμματα για μερικά από τα πιο δημοφιλή λειτουργικά συστήματα, επιτρέποντας τη διανομή του βασισμένου σε Python λογισμικού για χρήση σε αυτά τα περιβάλλοντα χωρίς να απαιτείται εγκατάσταση του διερμηνευτή της Python.

Η Python αναπτύσσεται ως ανοιχτό λογισμικό (open source) και η διαχείρισή της γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation. Ο κώδικας διανέμεται με την άδεια Python Software Foundation License η οποία είναι συμβατή με την GPL. Το όνομα της γλώσσας προέρχεται από την ομάδα άγγλων κωμικών Μόντυ Πάιθον.

Αρχικά, η Python ήταν γλώσσα σεναρίων που χρησιμοποιούνταν στο λειτουργικό σύστημα Amoeba, ικανή και για κλήσεις συστήματος.

Η Python 2.0 κυκλοφόρησε στις 16 Οκτωβρίου του 2000. Στις 3 Δεκεμβρίου 2008 κυκλοφόρησε η έκδοση 3.0 (γνωστή και ως py3k ή python 3000). Πολλά από τα καινούργια χαρακτηριστικά αυτής της έκδοσης έχουν μεταφερθεί στις εκδόσεις 2.6 και 2.7 που είναι προς τα πίσω συμβατές.

Η python 3 είναι ιστορικά η πρώτη γλώσσα προγραμματισμού που σπάει την προς τα πίσω συμβατότητα με προηγούμενες εκδόσεις ώστε να διορθωθούν κάποια λάθη που υπήρχαν σε προγενέστερες εκδόσεις και να καταστεί ακόμα πιο σαφής ο απλός τρόπος με τον οποίο μπορούν να γίνουν κάποια πράγματα

Η γλώσσα χρησιμοποιεί μεταγλωττιστή (compiler) για την δημιουργία του εκτελέσιμου κώδικα και σχετίζεται με τις γλώσσες προγραμματισμού Tcl, Perl, Scheme, Java και Ruby, καθώς και με την ABC η οποία υπήρξε η αρχική πηγή έμπνευσης για τη δημιουργία της.

Ένα από τα πιο απλά προγράμματα στην γλώσσα Python είναι η εμφάνιση ενός γραπτού αποτελέσματος (π.χ. Γεια σου, κόσμε!):

```
>>>print("Γεια σου, κόσμε!")
Γεια σου, κόσμε!
```

Ένα ιδιαίτερο χαρακτηριστικό της γλώσσας είναι η χρήση κενών διαστημάτων (whitespace) για τον διαχωρισμό των συντακτικών δομών που προγράμματος, σε αντίθεση με την πρακτική σε άλλες γλώσσες όπου για τον ίδιο σκοπό χρησιμοποιούνται ειδικά σύμβολα (πχ αγκύλες). Αυτό,

σε συνδυασμό με το ότι χρησιμοποιεί πλήρεις αγγλικές λέξεις στη θέση συμβόλων, καθιστούν τον κώδικα της Python ευανάγνωστο από όσους έχουν βασική γνώση των αγγλικών.

Για παράδειγμα, ο παρακάτω γεννήτορας (generator) `generate_primes` παράγει πρώτους αριθμούς:

```
from itertools import count
def generate_primes(stop_at=None):
  primes = []
  for n in count(2):
    if stop_at is not None and n > stop_at:
      return
    composite = False
    for p in primes:
      if not n % p:
        composite = True
        break
      elif p**2 > n:
        break
    if not composite:
      primes.append(n)
      yield n
```

Και για να καλέσουμε τη συνάρτηση, υπολογίζοντας τους πρώτους αριθμούς από το 1 στο 100:

```
for i in generate_primes():
  if i > 100: break
  print(i)
```

# 3 Οι αλγόριθμοι που χρησιμοποιήσαμε

## 3.1 Ο αλγόριθμος του Καίσαρα

Ο Κώδικας του Καίσαρα είναι μία από τις απλούστερες και πιο γνωστές τεχνικές κωδικοποίησης στην κρυπτογραφία. Είναι κώδικας αντικατάστασης στον οποίο κάθε γράμμα του κειμένου αντικαθίσταται από κάποιο άλλο γράμμα με σταθερή απόσταση κάθε φορά στο αλφάβητο. Για παράδειγμα, με μετατόπιση 3, το Α θα αντικαθιστούνταν από το Δ, το Β από το Ε, και ούτω καθεξής. Η μέθοδος πήρε το όνομά της από τον Ιούλιο Καίσαρα, ο οποίος την χρησιμοποιούσε στην προσωπική του αλληλογραφία.

Ο μετασχηματισμός μπορεί να αναπαρασταθεί με παράλληλη παράθεση δύο αλφαβήτων. Τα αλφάβητο κωδικοποίησης είναι το απλό αλφάβητο περιστραμένο δεξιά ή αριστερά κατά κάποιο αριθμό θέσεων. Για παράδειγμα ακολουθεί ένας κώδικας του Καίσαρα που χρησιμοποιεί αριστερή περιστροφή τριών θέσεων (η παράμετρος μετατόπισης, εδώ 3, χρησιμοποιείται ως κλειδί):

Απλό: ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ

Κώδικας: ΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩΑΒΓ

Όταν γίνεται κρυπτογράφηση, αναζητείται κάθε γράμμα της «απλής» γραμμής και γράφεται το αντίστοιχο γράμμα από την γραμμή του «κώδικα». Η αποκρυπτογράφηση γίνεται με την αντίστροφη φορά.

Κρυπτογραφημένο κείμενο: ΛΔΠΔΧΣΦ ΘΜΠΔΜ ΣΜ ΝΔΥΖΘΦ ΤΣΨ ΑΧΨΤΜΣΨΠΧΔΜ

Απλό κείμενο: θάνατος είναι οι κάργες που χτυπιούνται

Η κρυπτογράφηση μπορεί να αναπαρασταθεί με την χρήση αριθμητικής υπολοίπων αν πρώτα μετασχηματιστούν τα γράμματα σε αριθμούς, σύμφωνα με τον κανόνα, Α = 0, Β = 1,.... Ω = 23. Η κρυπτογράφηση ενός γράμματος x με μετατόπιση n μπορεί να περιγραφεί μαθηματικώς ως,

$$E_n(x) = (x + n) \mod 24$$

Η αποκρυπτογράφηση γίνεται αναλόγως,

$$D_n(x) = (x - n) \mod 24$$

(Υπάρχουν διαφορετικοί ορισμοί για την πράξη modulo. Στα παραπάνω το αποτέλεσμα βρίσκεται στο εύρος 0...23. Ήτοι, αν x+n ή x-n δεν βρίσκονται στο εύρος 0...23, αφαιρείται ή προστίθεται 24.) Η αντικατάσταση παραμένει η ίδια σε όλο το μήνυμα, έτσι ο κώδικας ταξινομείται ως μονοαλφαβητικής αντικατάστασης, σε αντίθεση με τους κώδικες πολυαλφαβητικής αντικατάστασης.

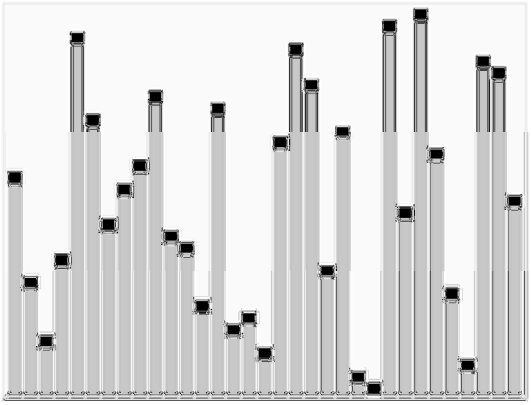
### 3.1.1 Ο αλγόριθμος σε python

```
import re


def caesar(plain_text, shift):
  cipherText = ''
  for ch in plain_text:
   stayInAlphabet = ord(ch) + shift
   if ch.islower():
    if stayInAlphabet > ord('z'):
     stayInAlphabet -= 26
    elif stayInAlphabet < ord('a'):
     stayInAlphabet += 26
   elif ch.isupper():
    if stayInAlphabet > ord('Z'):
     stayInAlphabet -= 26
    elif stayInAlphabet < ord('A'):
```

```
        stayInAlphabet += 26
    finalLetter = chr(stayInAlphabet)
    cipherText += finalLetter
  print(cipherText)
  return cipherText


selection = input ("kriptografisi/apokriptografisi ")
if selection == 'encrypt':
 plainText = input("Poio einai to keimeno? ")
 shift = (int(input("Bale to kleidi ")))%26
 caesar(plainText, shift)

else:
 plainText = input("Poio einai to keimeno? ")
 shift = ((int(input("Bale to kleidi ")))%26)*-1
 caesar(plainText, shift)
```

## 3.2 Ο αλγόριθμος Quicksort

Στην επιστήμη των υπολογιστών η γρήγορη ταξινόμηση (Quick-sort ) είναι ένας αλγόριθμος ταξινόμησης ο οποίος αναπτύχθηκε από τον Tony Hoare, που κατά μέσο όρος κάνει O(nlogn) συγκρίσεις για να ταξινομήσει n στοιχεία. Στην χειρότερη, σπάνια περίπτωση κάνει O(n^2) συγκρίσεις. Ο αλγόριθμος γρήγορης ταξινόμησης συχνά είναι γρηγορότερος από αντίστοιχους άλλους O(nlogn) αλγορίθμους και κατατάσσεται στους αλγορίθμους διαιρώ και βασιλεύω (όπου το πρόβλημα διασπάται σε μικρότερα προβλήματα και λύνεται το κάθε πρόβλημα ξεχωριστά).



Ο αλγόριθμος γρήγορης ταξινόμησης αναπτύχθηκε το 1960 από τον Tony Hoare την εποχή που αυτός ήταν επισκέπτης-φοιτητής στο Κρατικό Πανεπιστήμιο της Μόσχας. Εκείνη την εποχή εργαζόταν σε ένα πρότζεκτ σχετικό με μηχανική μετάφραση στο Κρατικό Εργαστήριο Φυσικής (National Physical Laboratory) στην Αγγλία. Υπήρχε την εποχή εκείνη ένα λεξικό μεταφρασμένων λέξεων από τα Ρωσικά στα Αγγλικά αποθηκευμένο αλφαβητικά μέσα σε μαγνητική ταινία το οποίο ήθελε να χρησιμοποιήσει για την μηχανική μετάφραση. Έτσι ανέπτυξε τον αλγόριθμο γρήγορης ταξινόμησης με σκοπό να ταξινομήσει αλφαβητικά τις λέξεις που έπρεπε να μεταφραστούν και να χρησιμοποιήσει τα δεδομένα της μαγνητικής ταινίας. Ο αλγόριθμος γρήγορης ταξινόμησης κέρδισε την γενικότερη αποδοχή και για παράδειγμα στη βιβλιοθήκη προγραμμάτων του Unix ενσωματώθηκε ως η κύρια συνάρτηση ταξινόμησης. Στην πρότυπη βιβλιοθήκη της γλώσσας προγραμματισμού C πήρε το όνομα qsort η οποία ενσωματώθηκε αργότερα και στην Java. Ο Robert Sedgewick έκανε διδακτορικό το 1975 στο Πανεπιστήμιο Stanford με αντικείμενο τον αλγόριθμο αυτό. Μέσα στη διδακτορική μελέτη έκανε εκτενή ανάλυση του αλγορίθμου αυτού και πρότεινε νέες βελτιώσεις
Έστω ότι έχουμε ένα πίνακα/λίστα με στοιχεία που θέλουμε να ταξινομήσουμε :
Αν ο πίνακας έχει 0 ή 1 στοιχεία δεν κάνουμε τίποτα (είναι ήδη ταξινομημένος) Αλλιώς αναδρομικά κάνουμε:
Επιλέγουμε ένα στοιχείο p (το οποίο ονομάζουμε pivot - άξονα) και το αφαιρούμε από την πίνακα/λίστα εισόδου.
Χωρίζουμε τον πίνακα/λίστα σε 2 μέρη: S1 και S2, όπου το S1 θα περιέχει όλα τα στοιχεία που είναι μικρότερα από το p και το S2 όπου περιέχονται όλα τα υπόλοιπα στοιχεία τα οποία είναι μεγαλύτερα ή ίσα με p.
Καλούμε τον αλγόριθμο αναδρομικά στο S1, παίρνουμε απάντηση στο T1 και στο S2 και παίρνουμε απάντηση στο T2.
Επιστρέφουμε τον πίνακα [T1, p, T2]

Με μορφή ψευδογλώσσας, ο αλγόριθμος ταξινόμησης το οποίος ταξινομεί στοιχεία από το p μέχρι το r σε ένα πίνακα A μπορεί να εκφραστεί ως

**quicksort(A, p, r):**
**if p < r:**
**q = partition(A, p, r)**

```
quicksort(A, p, q - 1)
quicksort(A, q + 1, p)

partition(A, p, r):
x = A[r]
i = p-1
for j=0 to r-1
if A[j] <= x
i = i+1
αντιμετάθεση A[i] με A[j]
αντιμετάθεση A[i+1] με A[r]
return i+1
```

### 3.2.1 Ο κώδικας σε python

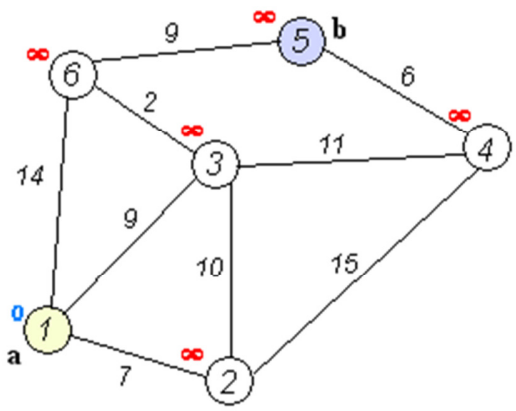
```
def quickSort(alist):
  quickSortHelper(alist,0,len(alist)-1)


def quickSortHelper(alist,first,last):
  if first < last:

    splitpoint = partition(alist,first,last)


    quickSortHelper(alist,first,splitpoint-1)
    quickSortHelper(alist,splitpoint+1,last)



def partition(alist,first,last):
  pivotvalue = alist[first]


  leftmark = first+1
  rightmark = last


  done = False
```

```
    while not done:

        while leftmark <= rightmark and alist[leftmark] <= pivotvalue:
            leftmark = leftmark + 1

        while alist[rightmark] >= pivotvalue and rightmark >= leftmark:
            rightmark = rightmark -1

        if rightmark < leftmark:
            done = True
        else:
            temp = alist[leftmark]
            alist[leftmark] = alist[rightmark]
            alist[rightmark] = temp

    temp = alist[first]
    alist[first] = alist[rightmark]
    alist[rightmark] = temp


    return rightmark

alist = [54,26,93,17,77,31,44,55,20]
quickSort(alist)
print(alist)
```

## 3.3 Ο αλγόριθμος Dijkstra

Ο αλγόριθμος του Dijkstra πήρε το όνομά του από τον Ολλανδό Έντσγκερ Ντάικστρα, ο οποίος τον επινόησε το 1956 και τον δημοσίευσε το 1959. Πρόκειται για έναν αλγόριθμο εύρεσης

συντομότερων διαδρομών (single-source shortest path problem) από κοινή αφετηρία σε έναν (κατευθυνόμενο ή μη) γράφο με μη αρνητικά βάρη στις ακμές. Ο αλγόριθμος του Dijkstra είναι άπληστος. Δηλαδή, σε κάθε βήμα επιλέγει την τοπικά βέλτιστη λύση, ώσπου στο τελευταίο βήμα συνθέτει μια συνολικά βέλτιστη λύση. Αν ο γράφος περιέχει αρνητικά βάρη, ο αλγόριθμος του Ντάικστρα δεν δίνει σωστό αποτέλεσμα. Για γράφους που μπορεί να έχουν αρνητικά βάρη στις ακμές, χρησιμοποιούνται πιο περίπλοκοι αλγόριθμοι, όπως αυτός των Bellman και Ford ή των Floyd-Warshall. Ο αλγόριθμος του Ντάικστρα είναι πλέον ευρέως διαδεδομένος και χρησιμοποιείται σε πολλές εφαρμογές. Χρήση του αλγόριθμου αυτού κάνει το πρωτόκολλο OSPF, το οποίο είναι το εσωτερικό πρωτόκολλο πύλης δικτύου του Διαδικτύου.
Έχουμε έναν γράφημα G(V,E), όπου V το σύνολο των κόμβων του και E το σύνολο των ακμών του. Επίσης, έχουμε μια συνάρτηση βάρους ορισμένη στις ακμές του γράφου. Αυτό σημαίνει ότι για να πάμε από έναν κόμβο του γράφου σε έναν άλλο, θα έχουμε κάποιο κόστος. Είναι σημαντικό, όπως θα φανεί παρακάτω, τα βάρη να μην είναι αρνητικά, επειδή διαφορετικά ο αλγόριθμος δεν δίνει σωστό αποτέλεσμα. Ο αλγόριθμος του Ντάικστρα βρίσκει τα μονοπάτια που πρέπει να ακολουθήσουμε από έναν κόμβο-αφετηρία προς τους υπόλοιπους, ώστε να έχουμε το λιγότερο δυνατό κόστος.




### 3.3.1 Ο αλγόριθμος σε βήματα:

Μια περισσότερο τυποποιημένη περιγραφή του αλγόριθμου είναι η παρακάτω, η οποία δείχνει τη λειτουργία του αλγόριθμου σε βήματα.
(1) Σημείωσε σε κάθε κόμβο μια ετικέτα απόστασης (d[*]) με τιμή 0 στον αρχικό κόμβο και τιμή άπειρο σε όλους τους υπόλοιπους. Επίσης, σημείωσε μια ετικέτα προηγούμενου κόμβου (prev[*]) και βάλε της την κενή τιμή για όλους τους κόμβους. Η ετικέτα αυτή χρειάζεται για τον υπολογισμό της ζητούμενης διαδρομής στο τέλος. (2) Σημείωσε όλους τους κόμβους μη-επεξεργασμένους (S = \emptyset). Ο τρέχων κόμβος είναι ο αρχικός.
(3) Για τον τρέχων κόμβο, εξέτασε όλους τους μη-επεξεργασμένους γείτονές του και υπολόγισε το συνολικό άθροισμα απόστασής τους από τον αρχικό κόμβο. Για παράδειγμα, αν ο τρέχων κόμβος έχει απόσταση 6 από τον αρχικό και ο γείτονας του τρέχοντος κόμβου, που εξετάζει αυτή τη στιγμή ο αλγόριθμος, έχει απόσταση 2 από τον τρέχων, το συνολικό άθροισμα απόστασης του γείτονα από τον αρχικό κόμβο είναι 6+2=8. Αν αυτή η απόσταση είναι μικρότερη από την ετικέτα απόστασης που είχε σημειωθεί, αντικατάστησέ τη με τη νέα

υπολογισμένη τιμή και σημείωσε τον τρέχων κόμβο στην ετικέτα προηγούμενου κόμβου.
(4) Όταν τελειώσεις με την εξέταση όλων των γειτόνων του τρέχοντος κόμβου, σημείωσέ τον ως επεξεργασμένο. Ένας επεξεργασμένος κόμβος δεν εξετάζεται ποτέ ξανά από τον αλγόριθμο. Η ετικέτα απόστασής της είναι η ελάχιστη και θα παραμείνει σταθερή.
(5) Ο επόμενος τρέχων κόμβος θα είναι ο μη-επεξεργασμένος κόμβος με τη μικρότερη ετικέτα απόστασης.
(6) Αν όλοι οι κόμβοι έχουν σημειωθεί ως επεξεργασμένοι, προχώρα στο επόμενο βήμα. Διαφορετικά, συνέχισε από το βήμα 3.
(7) Ξεκινώντας από τον κόμβο-προορισμό (ο οποίος είναι ο τελευταίος τρέχων κόμβος) εκτύπωσε τον κόμβο που αναγράφεται στην ετικέτα προηγούμενου κόμβου. Επανέλαβε μέχρι η ετικέτα προηγούμενο κόμβου που θα συναντήσεις να είναι άδεια.
Καλό είναι να αναφερθεί εδώ, ότι από τη στιγμή που υπάρχει η σημείωση 'επεξεργασμένο' δεν είναι απαραίτητη και η σημείωση 'μη-επεξεργασμένο' και το αντίστροφο. Για την υλοποίηση του αλγόριθμου μπορεί να επιλεγεί ένας από τους δύο τρόπους. Ο αλγόριθμος επιστρέφει τη συντομότερη διαδρομή ανάποδα, καθώς ξεκινά από τον προορισμό και εκτυπώνει κάθε φορά τον προηγούμενο. Είναι εύκολο όμως να επεξεργαστεί κανείς κατάλληλα αυτή τη σειρά και να την αναποδογυρίσει, ώστε να λάβει τη σωστή.

### 3.3.2 Ο αλγόριθμος σε Python

```
nodes = ('A', 'B', 'C', 'D', 'E', 'F', 'G')

distances = {

  'B': {'A': 5, 'D': 1, 'G': 2},

  'A': {'B': 5, 'D': 3, 'E': 12, 'F' :5},

  'D': {'B': 1, 'G': 1, 'E': 1, 'A': 3},

  'G': {'B': 2, 'D': 1, 'C': 2},

  'C': {'G': 2, 'E': 1, 'F': 16},

  'E': {'A': 12, 'D': 1, 'C': 1, 'F': 2},

  'F': {'A': 5, 'E': 2, 'C': 16}}


unvisited = {node: None for node in nodes} #using None as +inf

visited = {}

current = 'B'

currentDistance = 0
```

```
unvisited[current] = currentDistance


while True:
    for neighbour, distance in distances[current].items():
        if neighbour not in unvisited: continue
        newDistance = currentDistance + distance
        if unvisited[neighbour] is None or unvisited[neighbour] > newDistance:
            unvisited[neighbour] = newDistance
    visited[current] = currentDistance
    del unvisited[current]
    if not unvisited: break
    candidates = [node for node in unvisited.items() if node[1]]
    current, currentDistance = sorted(candidates, key = lambda x: x[1])[0]


print(visited)
```

# 4 Διαδραστικότητα σε python

Για την διαδραστικότητα σε python χρησιμοποιήσαμε το εργαλείο gistexec Που είναι ελεύθερο , με τον πηγαίο κώδικα https://github.com/rgbkrk/gistexec .