

ΣΧΟΛΗ ΑΝΘΡΩΠΙΣΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΠΑΙΔΑΓΩΓΙΚΟ ΤΜΗΜΑ ΔΗΜΟΤΙΚΗΣ ΕΚΠΑΙΔΕΥΣΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Ανάπτυξη του αλγοριθμικού τρόπου σκέψης με τη χρήση της γλώσσας προγραμματισμού Python**

Νεφέλη Κωστακοπούλου (Α.Μ.: 1049135)

Επιβλέπων καθηγητής: Παναγιωτακόπουλος Χρήστος

Ιούνιος 2019

Όταν αποφάσισα να εκπονήσω την παρούσα εργασία, έμπνευσή μου στάθηκε μία ιδέα ενός μυθιστορήματος επιστημονικής φαντασίας από το 1966 του Samuel R. Delany, το «Babel-17». Κεντρική ιδέα του βιβλίου είναι η κατασκευή της ομώνυμης γλώσσας Babel-17, η οποία «αναβαθμίζει» τον τρόπο σκέψης όλων όσοι τη μιλούν. «*Η Babel-17 είναι τόσο αναλυτική γλώσσα, ώστε σχεδόν σου εξασφαλίζει τεχνική υπεροχή απέναντι σε οποιαδήποτε κατάσταση αντιμετωπίζεις*», αναφέρει σε κάποιο σημείο ο πρωταγωνιστής. Αυτή ακριβώς την ιδέα προσπάθησα να πλησιάσω κι εγώ: να ενισχύσω και να βελτιώσω τον τρόπο που σκέφτονται οι μαθητές…

Στον Σ.Λ.

# Περίληψη

Η παρούσα εργασία πραγματεύεται την ένταξη της διδασκαλίας του προγραμματισμού στο Αναλυτικό Πρόγραμμα Σπουδών (ΑΠΣ) του Δημοτικού Σχολείου και το ρόλο της στην ανάπτυξη ή/και ενίσχυση της αλγοριθμικής σκέψης των μαθητών. Η επιλογή της γλώσσας προγραμματισμού με την οποία οι μαθητές έχουν την πρώτη τους επαφή με τον προγραμματισμό καθορίζει σε μεγάλο βαθμό τις πρώτες αναπαραστάσεις που δημιουργούν για τις αλγοριθμικές δομές και την στάση τους απέναντι στον προγραμματισμό. Η γλώσσα προγραμματισμού Python, καθώς είναι μια ανοικτού κώδικα γλώσσα εύκολα κατανοητή για τον αρχάριο προγραμματιστή και με προοπτικές εξέλιξης, η οποία κερδίζει όλο και περισσότερο έδαφος στον χώρο της εκπαίδευσης. Στην παρούσα εργασία επιλέχθηκε η Python όχι για την εκμάθηση της γλώσσας προγραμματισμού αυτής καθ’ αυτής, αλλά για την υποστήριξη δεξιοτήτων επίλυσης προβλημάτων από τους μαθητές μέσω αυτής. Στο πρώτο μέρος της εργασίας παρουσιάζονται οι βασικές αρχές και χαρακτηριστικά των γλωσσών προγραμματισμού γενικότερα και της Python ειδικότερα, καθώς και αναλυτικά η έννοια του αλγορίθμου. Επίσης, συζητείται η σημασία της διδασκαλίας του προγραμματισμού σε Α/θμια και Β/θμια εκπ/ση καθώς και μοντέλα ένταξής της στα ΑΠΣ. Στο δεύτερο μέρος γίνεται μια εισαγωγή στο βασικό λεξιλόγιο, τη σύνταξη και τις δομές της γλώσσας προγραμματισμού Python για τον εκπαιδευτικό καθώς και πρακτική εφαρμογή τους μέσα από δραστηριότητες με ενδεικτικό τρόπο επίλυσης και σκέψης, που βοηθούν τον εκπαιδευτικό να τα εντάξει ομαλά στην εκπαιδευτική διαδικασία. Τέλος, προτείνονται φύλλα εργασίας για την εύκολη εκμάθηση της γλώσσας Python από τους μαθητές, ώστε να είναι αργότερα σε θέση να τη χρησιμοποιήσουν με ευχέρεια ως γνωστικό εργαλείο για την επίλυση προβλημάτων και να ενισχυθεί ή/και να αναπτυχθεί ο αλγοριθμικός τρόπος σκέψης τους.

# Abstract

This thesis discusses the integration of teaching computer programming in Elementary School curriculum and its role in developing or/and enhancing students’ algorithmic thinking. The first programming language with which students have their first contact with programming determines the first representations they create for their algorithmic structures and their attitude to programming. The Python programming language is an open source evolving language easily understood by novice programmers popular in the field of education. In this thesis, Python was chosen to teach and learn not the programming language itself, but to support students’ problem-solving skills.In the first part of this thesis the basic principles and features of programming languages ​​in general and especially Python, as well as the concept of algorithm are presented. Furthermore, the importance of teaching of programming at elementary and secondary education as well as models of their integration in different countries’ curricula are being discussed. In the second part an introduction to the basic vocabulary, syntax and structures of the Python programming language aimed at teachers is given. A practical guide with programming activities, including an indicative way of thinking and solving, in order to help teachers to smoothly integrate programming into the educational process. Additionally, a series of worksheets are proposed to help students learn to code so as they can solve problems using Python programming language as a cognitive tool, targeting the foundations of programming: problem-solving, logic, and critical thinking, which are features of algorithmic thinking.

Περιεχόμενα

[Περίληψη 3](#_Toc12385605)

[Abstract 4](#_Toc12385606)

[Εννοιολογικοί προσδιορισμοί 6](#_Toc12385607)

[Εισαγωγή 7](#_Toc12385608)

[1o ΜΕΡΟΣ 9](#_Toc12385609)

[**1.** **ΤΠΕ στην εκπαίδευση** 9](#_Toc12385610)

[1.1 Υπολογιστική σκέψη – Αλγοριθμικός τρόπος σκέψης 12](#_Toc12385611)

[1.2 Θεωρίες μάθησης και γνωστική ανάπτυξη 14](#_Toc12385612)

[1.3 Ο προγραμματισμός ως γνωστικό εργαλείο 18](#_Toc12385613)

[**2.** **Αλγόριθμος και προγραμματισμός** 20](#_Toc12385614)

[2.1 Ορισμός 20](#_Toc12385615)

[2.2 Ιστορική αναδρομή του όρου και της έννοιας του αλγορίθμου 20](#_Toc12385616)

[2.3 Σχεδιασμός και ανάπτυξη αλγορίθμου 21](#_Toc12385617)

[2.4 Δημιουργία και αναπαράσταση αλγορίθμου 23](#_Toc12385618)

[2.5 Αλγόριθμος και δημιουργικότητα 25](#_Toc12385619)

[**3.** **Προγραμματισμός** 27](#_Toc12385620)

[3.1 Τι είναι προγραμματισμός 27](#_Toc12385621)

[3.1.1 Ιστορική αναδρομή 28](#_Toc12385622)

[3.1.2 Γλώσσες προγραμματισμού 30](#_Toc12385623)

[3.2 Η γλώσσα προγραμματισμού Python 31](#_Toc12385624)

[3.2.1 Ιστορική αναδρομή της Python 31](#_Toc12385625)

[3.2.2 Η Python στην Εκπαίδευση 32](#_Toc12385626)

[**4.** **Η επιστήμη των Υπολογιστών στην εκπαίδευση** 35](#_Toc12385627)

[4.1 Μοντέλα εισαγωγής της Επιστήμης των Υπολογιστών στην Εκπαίδευση 35](#_Toc12385628)

[4.2 Η ένταξη του προγραμματισμού στην Εκπαίδευση: πλεονεκτήματα και προκλήσεις 37](#_Toc12385629)

[**5.** **Σκοπός της εργασίας** 39](#_Toc12385630)

[**6.** **Προτάσεις για περεταίρω έρευνα** 40](#_Toc12385631)

[2o ΜΕΡΟΣ 41](#_Toc12385632)

[I. Εκπαιδευτικό υλικό για τον εκπαιδευτικό 41](#_Toc12385633)

[II. Γλωσσάρι 56](#_Toc12385634)

[III. Δραστηριότητες 58](#_Toc12385635)

[IV. Ενδεικτικά φύλλα εργασίας για τους μαθητές 74](#_Toc12385636)

[Βιβλιογραφία 117](#_Toc12385637)

# 

# Εννοιολογικοί προσδιορισμοί

**Τεχνολογίες της Πληροφορίας και των Επικοινωνιών - ΤΠΕ (Information and Communications technology - ICT):** Οι ΤΠΕ είναι οι μέθοδοι και τα προϊόντα της σύγχρονης τεχνολογίας που αναφέρονται στην συγκέντρωση, την  ηλεκτρονική κωδικοποίηση, την επεξεργασία και την ταξινόμηση της πληροφορίας, οποιαδήποτε και αν είναι η μορφή της (π.χ. κείμενο, ήχος, εικόνα, γράφημα, κτλ.). Το κέντρο των ΤΠΕ είναι ο ηλεκτρονικός υπολογιστής, ο οποίος είναι εμπλουτισμένος με δυνατότητες πολυμέσων και υπερμέσων που από την μία επιτρέπουν την επεξεργασία και την αποθήκευση κειμένου, εικόνας και ήχου και από την άλλη την μη γραμμική διάδοση της πληροφορίας με την χρήση υπερσυνδέσμων (links) (Buabeng-Andoh, 2012).

**Προγραμματισμός (Computer programming - coding):** Η διαδικασία σχεδιασμού και ανάπτυξης ενός εκτελέσιμου προγράμματος με σκοπό την αυτοματοποιημένη εκτέλεση εργασιών ή επίλυση κάποιου υπολογιστικού προβλήματος από έναν υπολογιστή. Ο προγραμματισμός περιλαμβάνει εργασίες όπως ανάλυση (ενός προβλήματος), παραγωγή αλγορίθμων, έλεγχο ορθότητας και ακρίβειας του αλγορίθμου (αποσφαλμάτωση - debugging) και εφαρμογή του αλγορίθμου σε μια γλώσσα προγραμματισμού (coding) (Gries, 1981).

**Επιστήμη υπολογιστών:** Μια γνωστική περιοχή που αναζητά την κατανόηση του κόσμου (φυσικού και τεχνητού) με χρήση υπολογιστικών όρων και υπολογιστικών διεργασιών. Μπορεί να θεωρηθεί ως η γνωστική περιοχή που συνδέεται με τη μελέτη, την ανάλυση, τον σχεδιασμό, την υλοποίηση των συστημάτων των υπολογιστών καθώς και των αρχών που τις διέπουν (Ψυχάρης & Καλοβρέκτης, 2018).

# Εισαγωγή

Από τα μέσα του 20ου αιώνα κι έπειτα όλο το φάσμα δραστηριότητας του ανθρώπινου πολιτισμού έχει εξαρτηθεί σε μικρότερο ή μεγαλύτερο βαθμό από τους Ηλεκτρονικούς Υπολογιστές (Η/Υ). Την τελευταία δεκαετία έχει επιτευχθεί τεράστια πρόοδος στον τομέα της Τεχνητής Νοημοσύνης (Artificial Intelligence) και του Machine Learning καθώς και στην ανάπτυξη των Κβαντικών Υπολογιστών (Quantum Computing). Όπως και με τη Βιομηχανική Επανάσταση, είναι αναμενόμενο ότι η πρόοδος των κοινωνιών θα στηρίζεται και θα εξαρτάται όλο και περισσότερο στους Η/Υ και την τεχνολογία, βελτιώνοντας δραματικά το βιοτικό επίπεδο των ανθρώπων. Σήμερα υπάρχουν πολλές διαφορετικές γλώσσες προγραμματισμού ειδικά σχεδιασμένες για να εφαρμόζονται σε συγκεκριμένους επιστημονικούς τομείς και πολλές άλλες γενικού σκοπού και χρήσης. Εκπαιδευτικά συστήματα πολλών αναπτυγμένων χωρών έχουν εντάξει τον Προγραμματισμό στο Αναλυτικό Πρόγραμμα του δημοτικού σχολείου, αναγνωρίζοντας τη σπουδαιότητά του τόσο ως γνωστικό εργαλείο αλλά και ως αναγκαίο εφόδιο για την προώθηση της καινοτομίας (Heintz, Mannila, Farnqvist, 2016).

Σκοπός της παρούσας εργασίας είναι η ανάδειξη της χρησιμότητας της διδασκαλίας του προγραμματισμού ως γνωστικό εργαλείο για την ενίσχυση της αλγοριθμικής σκέψης των μαθητών. Η επιλογή της γλώσσας προγραμματισμού με την οποία οι μαθητές θα έχουν την πρώτη τους επαφή με τον προγραμματισμό καθορίζει σε μεγάλο βαθμό τις πρώτες αναπαραστάσεις που θα δημιουργήσουν για τις αλγοριθμικές δομές και την στάση τους απέναντι στον προγραμματισμό (Kelleher & Pausch, 2005; McIver, 1996).

Για τον σκοπό αυτό επιλέχθηκε η γλώσσα προγραμματισμού Python, καθώς είναι μια ανοικτού κώδικα γλώσσα εύκολα κατανοητή για τον αρχάριο προγραμματιστή και με προοπτικές εξέλιξης. Αναπτύχθηκε εκτενές εκπαιδευτικό υλικό για τον εκπαιδευτικό, σύντομες οδηγίες για την αξιοποίηση της Python στην τάξη και ενδεικτικά φύλλα εργασίας με δραστηριότητες για τους μαθητές. Κατά τη διάρκεια των δραστηριοτήτων οι μαθητές μαθαίνουν σταδιακά πως να προγραμματίζουν (λεξιλόγιο, σύνταξη, τεχνικές λεπτομέρειες κ.α.) και ταυτόχρονα να επιλύουν ένα πρόβλημα κάνοντας χρήση των όσων έμαθαν.

Στόχος της εργασίας δεν είναι η εκμάθηση της γλώσσας προγραμματισμού αυτής καθ’ αυτής, αλλά η ανάπτυξη ή/και ενίσχυση των δεξιοτήτων επίλυσης προβλημάτων μέσω αυτής. Αυτό επιτυγχάνεται αφενός γιατί η έκφραση σε μια γλώσσα προγραμματισμού απαιτεί σαφή διατύπωση του αλγορίθμου επίλυσης και αφετέρου γιατί δεν υποδεικνύεται, όταν δεν είναι απαραίτητο, προς χάριν της δημιουργικότητας ένας συγκεκριμένος τρόπος επίλυσης.

# 1o ΜΕΡΟΣ

# **ΤΠΕ στην εκπαίδευση**

*«Technology is just a tool. In terms of getting the kids working*

*together and motivating them, the teaching is the most important.»*

*-Bill Gates*

Η εισαγωγή των ΤΠΕ στην εκπαίδευση αποτελεί μία πάγια ευρωπαϊκή πολιτική, που προωθείται από την Ευρώπη ήδη από τις αρχές της δεκαετίας του 90′, καθότι συμβάλλει στην βελτίωση της ποιότητας της εκπαιδευτικής διαδικασίας και της μάθησης. Ειδικότερα, αυτό που επιδιώκει η Ευρώπη με την εισαγωγή και την προώθηση των ΤΠΕ στην εκπαίδευση είναι: η βελτίωση της εκπαίδευσης και της κατάρτισης των εκπαιδευτικών, η ενεργητική εμπλοκή των μαθητών στην εκπαιδευτική διαδικασία, η ισότιμη πρόσβαση στην πληροφορία και στην γνώση, η ανάπτυξη μαθησιακών δεξιοτήτων, η δημιουργία «ανοικτών περιβαλλόντων» μάθησης, που ευνοούν την αλληλεπίδραση μεταξύ των εκπαιδευτικών και των μαθητών και η δημιουργία «ελκυστικών» σχολείων (Europa, 2017, European Commission, 2011).

Το Παιδαγωγικό Ινστιτούτο σχεδίασε το 1997 ένα “Ενιαίο Πλαίσιο Προγράμματος Σπουδών Πληροφορικής” [ΕΠΠΣΠ] που αφορά στην εισαγωγή και ένταξη των ΤΠΕ σε όλες τις βαθμίδες της εκπαίδευσης. Το ΕΠΠΣΠ προβλέπει ότι οι μαθητές που τελειώνουν το δημοτικό σχολείο πρέπει να είναι σε θέση “να περιγράφουν τα βασικά στοιχεία της αρχιτεκτονικής των υπολογιστών (μνήμη, επεξεργασία, περιφερειακά), να αναγνωρίζουν την κεντρική μονάδα και τις βασικές περιφερειακές συσκευές (πληκτρολόγιο, οθόνη, ποντίκι, εκτυπωτής) του υπολογιστή, να μπορούν να εξηγούν με απλά λόγια τη χρησιμότητά τους, να τις θέτουν σε λειτουργία και να τις χρησιμοποιούν, να εργάζονται με σχετική αυτονομία σε ένα γραφικό περιβάλλον εργασίας, να χρησιμοποιούν λογισμικό γενικής χρήσης για να εκφράζουν τις ιδέες τους με πολλούς τρόπους και μέσα (χρησιμοποιώντας εικόνες, ήχους, κείμενα κτλ.), να χρησιμοποιούν εφαρμογές πολυμέσων εκπαιδευτικού περιεχομένου και να έχουν κατακτήσει τις έννοιες της πλοήγησης σε ένα δίκτυο πληροφοριών και της αλληλεπίδρασης με ένα πληροφορικό σύστημα, να αναζητούν πληροφορίες από απλές βάσεις δεδομένων, να επικοινωνούν και να αναζητούν πληροφορίες χρησιμοποιώντας τον παγκόσμιο ιστό πληροφοριών, να αναφέρουν εφαρμογές της πληροφορικής στο σύγχρονο κόσμο, να αντιλαμβάνονται τον υπολογιστή, τις περιφερειακές συσκευές και το χρησιμοποιούμενο λογισμικό ως ενιαίο σύστημα.”

Το Νοέμβριο του 2003 παρουσίασε το “Διαθεματικό Ενιαίο Πλαίσιο Σπουδών Πληροφορικής” [ΔΕΠΠΣΠ] που τροποποίησε και διεύρυνε το ΕΠΠΣΠ. Σύμφωνα με το ΔΕΠΠΣΠ, “σκοπός της εισαγωγής της Πληροφορικής στο Νηπιαγωγείο και στο Δημοτικό Σχολείο είναι να εξοικειωθούν οι μαθητές και οι μαθήτριες με τις βασικές λειτουργίες του υπολογιστή και να έλθουν σε μια πρώτη επαφή με διάφορες χρήσεις του ως εποπτικού μέσου διδασκαλίας, ως γνωστικού – διερευνητικού εργαλείου και ως εργαλείου επικοινωνίας και αναζήτησης πληροφοριών στο πλαίσιο των καθημερινών σχολικών τους δραστηριοτήτων με τη χρήση κατάλληλου λογισμικού και ιδιαίτερα ανοικτού λογισμικού διερευνητικής μάθησης.”

Η χρήση του υπολογιστή, στο παραπάνω πλαίσιο, μπορεί να στραφεί γύρω από τέσσερις κεντρικούς άξονες:

**Γνωστικό - διερευνητικό εργαλείο:** χρήση ανοικτού λογισμικού διερευνητικής μάθησης για δημοτικό σχολείο. Το λογισμικό αυτό μπορεί να έχει τη μορφή αλληλεπιδραστικών πολυμέσων, προσομοίωσης, εκπαιδευτικού παιχνιδιού, μοντελοποίησης κλπ. και θα προσφέρει στους μαθητές τη δυνατότητα διερεύνησης πραγματικών ή φανταστικών καταστάσεων, αντίστοιχων του επιπέδου ωριμότητάς τους, διευκολύνοντας την ανάπτυξη της δημιουργικής και ανακαλυπτικής μάθησης. Ο υπολογιστής γίνεται μέσο για την ανάπτυξη δραστηριοτήτων και για την οργάνωση γνώσεων και δεξιοτήτων.

**Εποπτικό μέσο διδασκαλίας σε βασικά γνωστικά αντικείμενα:** αποτελεσματική χρήση του υπολογιστή με λογισμικό ευρείας χρήσης (π.χ. ζωγραφική, επεξεργασία κειμένου, λογισμικό φύλλο) που θα εντάσσεται στα πλαίσια της διδασκαλίας βασικών μαθημάτων: γλώσσα - γραπτή έκφραση, μαθηματικά, δημιουργία και ανάπτυξη δεξιοτήτων στις καλλιτεχνικές και τις συλλογικές δραστηριότητες.

**Εργαλείο επικοινωνίας και αναζήτησης πληροφοριών:** χρήση βάσεων δεδομένων για αναζήτηση στοιχείων, χρήση των δικτύων για επικοινωνία με άλλους μαθητές και για αναζήτηση πληροφοριών.

**Πληροφορικός αλφαβητισμός:** προσέγγιση των βασικών λειτουργιών του υπολογιστή: μνήμη, επεξεργασία της πληροφορίας, επικοινωνία, μέσα σε μια προοπτική τεχνολογικού αλφαβητισμού και αναγνώρισης των δυνατοτήτων της υπολογιστικής τεχνολογίας.

Βασικός σκοπός για τους μαθητές του Δημοτικού σχολείου είναι η κατανόηση των βασικών αρχών που διέπουν τη χρήση της υπολογιστικής τεχνολογίας σε σημαντικές ανθρώπινες ασχολίες, στα πλαίσια ποικίλων σχολικών δραστηριοτήτων που αναπτύσσουν με τον υπολογιστή. Σημαντικός παράγοντας αποτελεί το γεγονός πως οι μαθητές σε αυτήν την ηλικία εξοικειώνονται με τον υπολογιστή χωρίς ιδιαίτερη προσπάθεια. Οι επιμέρους σκοποί είναι να προσεγγίσουν οι μαθητές βασικές έννοιες Πληροφορικής, να γνωρίσουν τον υπολογιστή και τις επιμέρους μονάδες του κατανοώντας τη χρησιμότητά τους και αναγνωρίζοντας της δυνατότητες της υπολογιστικής τεχνολογίας, να εξοικειωθούν με τη χρήση λογισμικού γενικού σκοπού, όπου ο υπολογιστής εντάσσεται στα πλαίσια της διδασκαλίας βασικών γνωστικών αντικειμένων, να μπορούν να αναζητήσουν πληροφορίες στο Διαδίκτυο ή σε άλλες πηγές πληροφοριών, να συνεργαστούν για την εκτέλεση ομαδικών δραστηριοτήτων, να συζητήσουν και να προβληματιστούν.

Συγκεκριμένα για την Στ’ δημοτικού προβλέπεται, εκτός των άλλων, η ενασχόληση των μαθητών με «μια απλή γλώσσα προγραμματισμού (Logo like) για τον έλεγχο και τον προγραμματισμό του υπολογιστή». Η παιδαγωγική αξία των γλωσσών προγραμματισμού σε όλες τις βαθμίδες εκπαίδευσης έχει βρεθεί επανειλημμένα από μεγάλης κλίμακας εμπειρικές μελέτες (Singh and Zwirner, 1996). Η χρήση γλωσσών προγραμματισμού στην εκπαιδευτική διαδικασία δεν είναι σκοπός αλλά μέσο. Στόχος της ενασχόλησης με τον προγραμματισμό δεν είναι η διδαχή και η εκμάθηση κάποιου συγκεκριμένου προγραμματιστικού περιβάλλοντος, ούτε η καλλιέργεια προγραμματιστικών δεξιοτήτων από τη μεριά των μαθητών. ∆εν αποσκοπεί στη λεπτομερειακή εξέταση της δομής, του ρεπερτορίου και των συντακτικών κανόνων κάποιας γλώσσας προγραμματισμού. ∆εν προτίθεται να επιχειρήσει να δημιουργήσει προγραμματιστές. Βασικός σκοπός του, όπως και όλων των δραστηριοτήτων στις οποίες εμπλέκονται οι μαθητές, δεν είναι μόνο η απόκτηση γνώσεων αλλά και o προβληματισμός σχετικά με διάφορα θέματα και η καλλιέργεια δεξιοτήτων όπως είναι η δεξιότητα επίλυσης προβλημάτων, η κριτική σκέψη και η οργανωμένη έκφραση.

Με βάση τα παραπάνω, μπορούμε να πούμε πως ο προγραμματισμός είναι ένα εργαλείο δόμησης της σκέψης, ένα εργαλείο που ενισχύει τον αλγοριθμικό τρόπο σκέψης των μαθητών.

## 1.1 Υπολογιστική σκέψη – Αλγοριθμικός τρόπος σκέψης

*«Computers are good at following instructions,*

*but not at reading you mind.»*

*-Donald E. Knuth*

Η Υπολογιστική Σκέψη (ΥΣ) είναι ένα σύνολο δεξιοτήτων που βρίσκουν εφαρμογή σε ένα ευρύ φάσμα προβλημάτων της καθημερινής ζωής. Η J. Wing (2006), καθηγήτρια του τμήματος Υπολογιστικής Σκέψης του Πανεπιστημίου Carnegie Mellon στο Πιτσμπουργκ, σε άρθρο της όρισε την ΥΣ ως «τις όψεις σχεδιασμού συστημάτων, επίλυσης προβλημάτων και κατανόησης της ανθρώπινης συμπεριφοράς». Ισχυρίστηκε ότι το να καταλαβαίνει κανείς τον κόσμο του δίνει τη δυνατότητα να εντοπίζει τα προβλήματα και να συνεισφέρει στην εύρεση λύσεων σε αυτά. Με τον όρο ΥΣ νοείται ένα είδος αναλυτικής σκέψης.

Η ΥΣ αποτελεί μια προσέγγιση επίλυσης προβλημάτων, σχεδιασμού συστημάτων και κατανόησης της ανθρώπινης συμπεριφοράς, η οποία βασίζεται σε θεμελιώδεις έννοιες προγραμματισμού (Wing, 2008). Περιλαμβάνει τη συλλογιστική σε πολλαπλά επίπεδα αφαίρεσης, τον προσδιορισμό, την κατανόηση και την επίλυση ενός προβλήματος με την χρήση της υπόθεσης και της έρευνας, την κατασκευή μοντέλου και τον πειραματισμό για τη δοκιμή της υπόθεσης και την εκτίμηση των αποδείξεων (Bransford & Donovan, 2005∙ Werner, Hanks, & McDowell, 2004). Πρόσθετα χαρακτηριστικά της είναι η δημιουργία αντικειμένων και η κατανόηση και η εφαρμογή αυτοματισμού. Αν και η έννοια της ΥΣ αντιπροσωπεύει μια γνωστική διαδικασία, υπάρχουν ερευνητικές εργασίες που θεωρούν ότι θα πρέπει να ενταχθούν σε αυτήν και άλλοι «τύποι» σκέψης όπως η λογική σκέψη, η αλγοριθμική σκέψη, η «σκέψη των μηχανικών» και η μαθηματική σκέψη (Ψυχάρης & Καλοβρέκτης 2018, Selby & Woollard 2013).

Η ΥΣ είναι ένα σύνολο γνωστικών δεξιοτήτων και διαδικασιών επίλυσης προβλημάτων. Κοινά αποδεκτά στοιχεία της ΥΣ είναι τα εξής (Ψυχάρης & Καλοβρέκτης, 2018):

* Εντοπισμός προτύπων (μοτίβα) για να αντιπροσωπεύσουν το πρόβλημα
* Λογική οργάνωση και ανάλυση δεδομένων
* Τμηματοποίηση (Decomposition) που αφορά στην αποσύνθεση του προβλήματος σε μικρότερα, γνωστά προβλήματα
* Δομοστοιχείωση (Modularity), η οποία αφορά στην προσέγγιση του προβλήματος με τεχνικές προγραμματιστικής σκέψης (δομή επανάληψης, συμβολική αναπαράσταση, λογικές πράξεις)
* Δημιουργία μιας σειράς πράξεων με σκοπό την επίλυση του προβλήματος (αλγοριθμική σκέψη)
* Εύρεση και ανάλυση πιθανών λύσεων και επιλογή της πιο αποτελεσματικής λύσης
* Γενίκευση της διαδικασίας για χρήση σε ένα πιο ευρύ φάσμα προβλημάτων

Στην εκπαίδευση, η ΥΣ έχει συνδεθεί άμεσα με τα Μαθηματικά και τους κλάδους που σχετίζονται με αυτά (Science, Technology,  Engineering, Mathematics - STEM). Οι καθηγητές των κλάδων αυτών ωθούν τους μαθητές να αναπτύξουν ικανότητες ΥΣ μέσω της εξάσκησης σε δεξιότητες επίλυσης προβλημάτων χρησιμοποιώντας τα μαθηματικά και, αντιστρόφως, χρησιμοποιώντας υπολογιστικά εργαλεία ενισχύουν τις μαθηματικές δεξιότητες των μαθητών. Με πιο απλά λόγια, η καλλιέργεια μαθηματικών δεξιοτήτων οδηγεί στη βελτίωση της ΥΣ και το αντίστροφο. Η Wing (2007) δηλώνει ότι η Επιστήμη των Υπολογιστών στηρίζεται στα Μαθηματικά και ο Sussman (NRC, 2010) επιβεβαιώνει ότι η μαθηματική σκέψη «περιστρέφεται» γύρο από αφηρημένες δομές όπως και η Υπολογιστική σκέψη. Οι Zhang και Luo (2012) προτείνουν την Υπολογιστική Σκέψη ως «ολοκλήρωση» της μαθηματικής σκέψης και της «σκέψης των μηχανικών».

Η Υπολογιστική Σκέψη είναι άρρηκτα συνδεδεμένη με τον προγραμματισμό, συνεπώς και με τα Μαθηματικά. Αυτό δεν είναι παράλογο, αν σκεφτεί κανείς πως κατά τη διαδικασία του προγραμματισμού ο χρήστης καλείται να αναγνωρίσει και να χειριστεί μοτίβα αλλά και το ότι η ανάπτυξη ενός άρτιου λογισμικού απαιτεί χρήση κατάλληλων πράξεων.

## 1.2 Θεωρίες μάθησης και γνωστική ανάπτυξη

*«Teaching means creating situations*

*where structures can be discovered.»*

*-Jean Piaget*

Οι θεωρίες μάθησης που μπορούν να υποστηρίξουν την ένταξη της μάθησης προγραμματισμού στην εκπαίδευση, άρα και τον χαρακτηρισμό του ως γνωστικό εργαλείο, είναι η κονστρουκτιβιστική θεωρία (Piaget, 1954), η κονστρακσιονιστικη θεωρία (Papert, 1980), η κοινωνικοπολιτισμική θεωρία (Vygotsky, 1978), η θεωρία της «Ανακαλυπτικής Μάθησης» (Bruner, 1966).

Η θεωρία του Piaget (κονστρουκτιβισμός – constructivism) παρέχει ένα πλαίσιο για την κατανόηση του πως τα παιδιά σκέφτονται και πράττουν σε διαφορετικά επίπεδα της ανάπτυξής τους. Σύμφωνα με τον Piaget, τα παιδιά σκέφτονται ανάλογα με τις εκάστοτε ανάγκες και δυνατότητές τους. Οι αλλαγές που σχετίζονται με τη σύλληψη ή αντίληψη εννοιών από τα παιδιά στα διαφορετικά επίπεδα ανάπτυξης αναδύονται ως αποτέλεσμα της δράσης στον κόσμο, δηλαδή την εμπειρία, σε συνδυασμό με μια σειρά από κρυφές διαδικασίες, που επιστρατεύονται για την εξισορρόπηση επιφανειακών διαταραχών (Carey, 1987, Kuhn). Αυτό σημαίνει πως το να υιοθετήσει κανείς ένα καινούριο σύστημα αξιών ή πεποιθήσεων απαιτεί περισσότερα από μία απλή έκθεση σε μία θεωρία, απαιτεί εμπλοκή σε αυτή τη θεωρία. Εντάσσοντας το παραπάνω στην εκπαιδευτική πραγματικότητα, ο μαθητής κατασκευάζει ενεργά τη γνώση μέσα στη σχολική τάξη, δεν τη δέχεται παθητικά μέσα από τα βιβλία ή τις διαλέξεις του δασκάλου.

Μια τέτοια οπτική της εκπαιδευτική διαδικασία αναδεικνύει τα εξής: *Πρώτον*, τα παιδιά δεν είναι σε θέση να αντιληφθούν τη διδασκαλία αν αυτή δεν ανταποκρίνεται σε γνώση και εμπειρία από την καθημερινή τους ζωή(Ackermann, 2001). *Δεύτερον*, η γνώση δεν είναι πληροφορίες που προσλαμβάνονται και ύστερα κωδικοποιούνται, απομνημονεύονται, ανασύρονται από τη μνήμη και εφαρμόζονται. Αντίθετα, πρόκειται για μια εμπειρία που μπορεί να αποκτηθεί μέσω της αλληλεπίδρασης με τον κόσμο, τους ανθρώπους και τα πράγματα (Ackermann, 2001). *Τρίτον*, τα παιδιά αντιστέκονται στην ιδέα της αλλαγής των αντιλήψεών τους (Ackermann, 2001), αλλά μία θεωρία μάθησης οφείλει να λάβει υπόψη αυτόν τον παράγοντα και να αναδύσει, μέσω σωστών πρακτικών, από τα ίδια τα παιδιά μια καλύτερη αντίληψη για τον κόσμο.

Ο προγραμματισμός μπορεί να υποστηρίξει τις τρεις παραπάνω διατυπώσεις της κονστρουκτιβιστικής μάθησης. Οι μαθητές έρχονται καθημερινά σε επαφή με Η/Υ, και έμμεσα με τον προγραμματισμό τους, οπότε το θέμα τους αφορά άμεσα. Ταυτόχρονα, τα παιδιά αλληλεπιδρούν με το προς μάθηση αντικείμενο, καθώς κατασκευάζουν αλγορίθμους, τους δοκιμάζουν, κάνουν λάθη και μαθαίνουν από αυτά, βελτιώνουν τις πρακτικές τους. Επιπλέον, τόσο ο ίδιος ο Η/Υ όσο και οι προοπτικές που δημιουργούνται μέσα από τον προγραμματισμό του αποτελούν από μόνα τους πόλους έλξης για τους μαθητές. Ιδιαίτερα αν ενταχθεί ομαλά στην εκπαιδευτική διαδικασία, ο προγραμματισμός είναι δυνατόν να δώσει στους μαθητές τις δυνατότητες και τα κίνητρα να αλλάξουν ή/και να αναπτύξουν τον τρόπο που αντιλαμβάνονται τα πράγματα.

Η θεωρία του Piaget πραγματεύεται το πως ο μαθητής κατασκευάζει τη γνώση ατομικά με βάση τις αντιληπτικές του ικανότητες σε κάθε στάδιο ανάπτυξης. Η θεωρία του Papert (κονστρακσιονισμός - constructionism), λαμβάνει υπόψη την κονστρουκτιβιστική θεωρία οικοδόμησης της γνώσης εσωτερικά από το άτομο και την επεκτείνει για να επικεντρωθεί στην στήριξη και ενίσχυση των εσωτερικών κατασκευών, μεταξύ άλλων, από τη χρήση Η/Υ και ρομποτικής. Κατά τον Papert (1991), ο κονστραξιονισμός αναδεικνύει τη θεωρία κατασκευής γνωστικών δομών του κονστρουκτιβισμού αρχικά μέσω της προοδευτικής εσωτερικοποίησης των δράσεων και ύστερα εισάγοντας την ιδέα ότι αυτή η εσωτερικοποίηση συμβαίνει στο πλαίσιο της προσπάθειας του ατόμου να κατασκευάσει μια άποψη για το πως λειτουργεί ο κόσμος γύρω του. Μέσω της προσέγγισής του προκύπτει ότι οι ιδέες πραγματώνονται και αλλάζουν όταν εκφράζονται μέσα από διαφορετικά μέσα, σε διαφορετικές συνθήκες, από διαφορετικούς ανθρώπους. Ο Papert προέβαλε μία μάθηση που καθοδηγείται από τους ίδιους τους μαθητευόμενους μέσω μιας διαδικασίας ανάπτυξης των δικών τους εργαλείων και πρακτικών που πιστεύουν ότι άπτονται περισσότερο στις ανάγκες της έρευνάς τους (Ackermann, 2001). Πίστευε πως η προβολή των συναισθημάτων και των ιδεών μας είναι το κλειδί της μάθησης, αφού το να τις επικοινωνούμε με άλλους προϋποθέτει να είναι ξεκάθαρα διατυπωμένες ενώ, ταυτόχρονα, αυξάνονται οι πιθανότητες να γίνουν πράξη.

Βασικοί στόχοι του κονστραξιονισμού (Papert, 1991∙ Resnick, 1994) είναι: *1)* η επίλυση προβλημάτων μέσω της χρήσης και ανάπτυξης πραγματικών και μη πραγματικών αντικειμένων, *2)* οργάνωση της σκέψης, μέσω της χρήσης αλγορίθμων και εντολών στο πλαίσιο μιας γλώσσας προγραμματισμού, *3)* η κοινωνικοποίηση, μέσω της συνεργασίας των μαθητών για την υλοποίηση ενός project, *4)* η διαθεματικότητα, η απόκτηση δηλαδή γνώσεων που αφορούν και άλλες εκφάνσεις της καθημερινότητας (Baron, 1994∙ Kafai & Resnick, 1996).

Οι στόχοι αυτοί είναι δυνατόν να υλοποιηθούν μέσω του προγραμματισμού, καθώς οι μαθητές επιλύουν προβλήματα αναπτύσσοντας αλγορίθμους που αντανακλούν το γνωστικό υπόβαθρο και τις ανάγκες του καθενός ξεχωριστά. Ταυτόχρονα, ωθούνται, λόγω της αυστηρής δομής των γλωσσών προγραμματισμού, στην τυποποίηση της σκέψης τους και στη σαφή διατύπωση των ιδεών τους, ενώ κοινωνικοποιούνται μέσω της ανταλλαγής ιδεών για την επίλυση ενός προβλήματος. Η όλη διαδικασία συμβάλλει στη διαθεματικότητα της μάθησης, αφού μπορεί να χρησιμοποιηθεί για τη διδασκαλία μαθηματικών, φυσικής, στατιστικής κ.α., ενώ μπορεί να συμβάλει και στην ανάπτυξη δεξιοτήτων επίλυσης προβλημάτων ακόμα και της καθημερινής ζωής.

Με την σημασία της εξωτερικοποίησης των ιδεών ως μέσο όξυνσης του μυαλού ασχολήθηκε και ο Vygotsky, ο οποίος ανέπτυξε τη θεωρία του κοινωνικού κονστρουκτιβισμού. Ασχολήθηκε με το κατά πόσο τα χαρακτηριστικά της κουλτούρας – τα εργαλεία, η γλώσσα και οι άνθρωποι – μπορούν να αποτελέσουν πηγή πληροφόρησης για τις γνωστικές δυνατότητες του ατόμου. Υποστήριζε ότι, κατά την εμπλοκή των ατόμων σε μια δραστηριότητα, οι πιο έμπειροι καθοδηγούν, αρχικά, εκείνους που μαθαίνουν. Με το πέρας του χρόνου, οι μαθητευόμενοι εξοικειώνονται με τη διαδικασία και αποκτούν γνώσεις και δεξιότητες που τους επιτρέπουν να συμμετέχουν όλο και περισσότερο στις ομαδικές δράσεις. Αυτή η συμμετοχή σε κοινές δράσεις συντελεί στην εσωτερικοποίηση των αποτελεσμάτων και τελικά στην απόκτηση ουσιαστικής γνώσης.

Ο Vygotsky (1978) εισήγαγε τη θεωρία «*Ζώνης Επικείμενης Ανάπτυξης*», η οποία αντιπροσωπεύει την απόσταση ανάμεσα σε αυτό που μπορεί να κάνει κανείς μόνος του ως αποτέλεσμα των ψυχικών του ικανοτήτων (ενδοψυχική λειτουργία) και σε αυτό που μπορεί να επιτύχει υπό την εποπτεία των ενηλίκων ή των περισσότερο ικανών συνομιλήκων (διαψυχική λειτουργία). Πρόκειται για εκείνες τις γνώσεις και τις δεξιότητες που το άτομο δεν είναι ακόμα ικανό να καταλάβει ή να πράξει μόνο του, αλλά μπορεί να κατακτήσει με την καθοδήγηση άλλων. Μέσα από τη συνεργασία και την αλληλεπίδραση ώριμων αναπτυξιακά μελών με λιγότερο ώριμα μέλη, τα παιδιά είναι σε θέση να αναπτύξουν μια κοινή οπτική όσον αφορά το περιβάλλον τους και να ορίσουν κοινές έννοιες, απαραίτητες για την κοινωνική τους ένταξη και ανέλιξη.

Η επιστήμη των υπολογιστών μπορεί να υποστηρίξει την εισαγωγή ΖΕΑ σε ομάδες μαθητών, αφού συνεργάζονται μεταξύ τους για την οικοδόμηση γνώσης (Crook, 1998). Στην πιο σύγχρονη βιβλιογραφία, η ΖΕΑ αναφέρεται ως «πλαίσιο στηρίγματος» (Ράπτης & Ράπτη, 2007 · Κόμης, 2004), όπου ο εκπαιδευτικός είναι εκείνος που λειτουργεί ως το μέσο του μαθητή για να κατανοήσει όσα του είναι απαραίτητα για να αναπτυχθεί γνωστικά και να αναπτύξει την προσωπικότητά του.

Ο Brunner επηρεασμένος από τους Piaget, Vygotsky και Maslow, πίστευε ότι η γνώση προκύπτει από την πράξη του μαθητή, είναι, όμως, στη βάση της κοινωνική και είναι πάντα σε στενή εξάρτηση με τα κίνητρά του. Εισήγαγε τη θεωρία της «*Ανακαλυπτικής Μάθησης*» (Brunner, 1966), σύμφωνα με την οποία οι μαθητές αναπτύσσουν τις γνώσεις και τις δεξιότητές τους μέσω του πειραματισμού και της πρακτικής. Η ανακάλυψη ως μορφή επίλυσης προβλημάτων δεν συνίσταται στην χωρίς επίβλεψη αυτενέργεια των μαθητών (Klarh & Simon, 1999).

Η «*Ανακαλυπτική μάθηση*» είναι μια διδακτική προσέγγιση, η οποία ενέχει ελάχιστη καθοδήγηση (*καθοδηγούμενη ανακάλυψη*), απαιτεί όμως ένα είδος κατεύθυνσης, κατά την οποία ο εκπαιδευτικός οργανώνει δραστηριότητες όπου οι μαθητές αναζητούν, χειρίζονται, εξερευνούν και διερευνούν τα δεδομένα προκειμένου να ολοκληρώσουν την δραστηριότητα ή να επιλύσουν το πρόβλημα (σπειροειδής οργάνωση) (Shunk, 2010). Οι μαθητές έτσι αποκτούν νέες γνώσεις, συναφείς με το εκάστοτε πεδίο και γενικές δεξιότητες επίλυσης προβλημάτων όπως η διατύπωση κανόνων, ο έλεγχος υποθέσεων και η συλλογή πληροφοριών (Bruner, 1966). Έτσι ο μαθητής οφείλει να έρχεται αντιμέτωπος με προβληματικές καταστάσεις, το αναλυτικό πρόγραμμα να οργανώνεται σε σπειροειδή μορφή και ο εκπαιδευτικός οφείλει να έχει ρόλο εμψυχωτή, διευκολυντή και συντονιστή στη διαδικασία της μάθησης (Bruner, 1997).

Ο προγραμματισμός βασίζεται στην ιδέα της *ανακαλυπτικής μάθησης* που θέλει τον άνθρωπο να οικοδομεί τις γνώσεις του και, έτσι, ωθεί τους μαθητές να διερευνήσουν και να συγκρίνουν πληροφορίες (κατανόηση προβλήματος), υποστηρίζουν την αναπαράσταση της γνώσης (δημιουργία αλγορίθμου) σε 3 διαστάσεις (πραξιακή, εικονιστική, συμβολική), εμπλέκουν τα παιδιά σε διαδικασίες επίλυσης προβλημάτων και επιδιώκουν τη συνεργατική μάθηση.

## Ο προγραμματισμός ως γνωστικό εργαλείο

*«The idea behind digital computers may be explained by saing that*

*these machines are intended to carry out any operations which*

*could be done by a human computer.»*

*-Alan Turing*

Ως «*εργαλείο*» ορίζεται μια συσκευή, μια διάταξη ή ένας μηχανισμός ο οποίος χρησιμοποιείται για την εκτέλεση ή διευκόλυνση εργασιών (Μπαμπινιώτης, 2002). Το εργαλείο δεν είναι απαραίτητα ένα φυσικό αντικείμενο, αλλά ενδέχεται να είναι ένα σύστημα συμβόλων για έκφραση και επικοινωνία, μια στρατηγική μέσω της οποίας μετασχηματίζεται η γνώση, μια εφαρμογή λογισμικού για τη διαχείριση και τη δημιουργία συμβολικού υλικού (Salomon, 1993). Το εργαλείο εμπεριέχει τους σκοπούς για τους οποίους έχει σχεδιαστεί, τους τρόπους με τους οποίους χρησιμοποιείται, τις τεχνικές που είναι απαραίτητες για τη χρήση του, τις γνώσεις και δεξιότητες που απαιτεί η χρήση του.

Ως «*γνωστικά εργαλεία*» (cognitive tools) αναφέρονται οι τεχνολογίες που υποστηρίζουν γνωστικές διεργασίες, όπως η σκέψη, η επίλυση προβλημάτων, η μάθηση. Τέτοια εργαλεία είναι ο γραπτός λόγος, η μαθηματική σημειογραφία, οι Τεχνολογίες της Πληροφορίας και των Επικοινωνιών (Reeves et al. 1997).

Στο πλαίσιο των ΤΠΕ, ως γνωστικό εργαλείο μπορεί να χρησιμοποιηθεί ο προγραμματισμός, καθώς μέσω αυτού είναι δυνατόν, με τον κατάλληλο σχεδιασμό, ο μαθητής να εμπλακεί σε διεργασίες με συγκεκριμένους διδακτικούς στόχους και σε νοητικές διεργασίες υψηλού επιπέδου (Μικρόπουλος, 2006), όπως η δημιουργία αλγορίθμων για την επίλυση ενός προβλήματος.

Η κατάλληλη και επαρκής αναπαράσταση ενός προβλήματος θεωρείται ως το κλειδί για την επίλυσή του (Jonassen, 2003). Υπό αυτή την έννοια, ο προγραμματισμός είναι το γνωστικό εργαλείο μέσω του οποίου ο μαθητής αναπαριστά τόσο τα δεδομένα του ζητούμενου προβλήματος, με τον ορισμό των κατάλληλων μεταβλητών και την εισαγωγή κατάλληλων δεδομένων, όσο και τη διαδικασία που είναι απαραίτητη για την επίλυσή του (δημιουργία αλγορίθμου).

Η παιδαγωγική αξιοποίηση των ΤΠΕ, και συνεπώς του προγραμματισμού, στο πλαίσιο των γνωστικών εργαλείων, όπου δίνεται έμφαση σε διεργασίες κατανόησης, σχεδιασμού, αναπαράστασης, αναστοχασμού και επίλυσης προβλημάτων, προϋποθέτει ένα πλαίσιο για τη σχεδίαση μαθησιακών δραστηριοτήτων για τη δημιουργία κατάλληλης κοινωνικής και διδακτικής προσέγγισης αλλά και για την προώθηση της ενεργού συμμετοχής των μαθητών. Στο παραπάνω πλαίσιο αφορούν οι παρακάτω αρχές (Μικρόπουλος, 2006):

1. Παροχή εργαλείων και δομών που διευκολύνουν τον μαθητή να χρησιμοποιεί στο μέγιστο βαθμό την ευφυΐα και τις γνώσεις του (Reusser, 1993).
2. Να δίνεται στον μαθητή η αίσθηση του ελέγχου των διεργασιών και της επίλυσης του προβλήματος από τον ίδιο μέσα από δραστηριότητες σχεδιασμού, αναπαράστασης, αναστοχασμού, λήψης αποφάσεων.
3. Να επιτρέπεται στον μαθητή να δημιουργεί και να εκφράζει τα νοητικά του μοντέλα.
4. Να παρέχονται αποτελεσματικά εργαλεία για τη δημιουργία αναπαραστάσεων.
5. Να επεκτείνεται η χρήση τους σε μια υποστηρικτική δομή συνεργατικής μάθησης.

# **Αλγόριθμος και προγραμματισμός**

*«The modern meaning for algorithm is quite similar to that of*recipe,

process, method, technique, procedure, routine, rigmarole,

*except that the word “algorithm” connotes something just a little different.»*

*-Donald E. Knuth*

## 2.1 Ορισμός

Στα Μαθηματικά και στην Επιστήμη των Υπολογιστών, ως *αλγόριθμος* ορίζεται μια σαφής και αυστηρά καθορισμένη σειρά ενεργειών εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος ή μιας σειράς προβλημάτων (Παναγιωτακόπουλος, 2013). Οι αλγόριθμοι αφορούν τόσο στην εκτέλεση υπολογισμών (calculation) όσο και στην επεξεργασία δεδομένων (data processing). [Αλγόριθμος](http://el.wikipedia.org/wiki/%CE%B1%CE%BB%CE%B3%CF%8C%CF%81%CE%B9%CE%B8%CE%BC%CE%BF%CF%82) μπορεί να είναι μια συνταγή μαγειρικής ή η βήμα προς βήμα περιγραφή της λύσης ενός μαθηματικού προβλήματος (Αράπογλου κ.α., 2006).

Σε ένα πολύ γενικό πλαίσιο ορισμού, λοιπόν, ένας αλγόριθμος είναι η βήμα προς βήμα αυστηρά καθορισμένη διαδικασία για την επίλυση ενός συγκεκριμένου προβλήματος.

## 2.2 Ιστορική αναδρομή του όρου και της έννοιας του αλγορίθμου

Η λέξη «αλγόριθμος» έχει τις ρίζες της στη λατινική μορφή του ονόματος του μαθηματικού, αστρονόμου, γεωγράφου και λόγιου Muhammad ibn Musa al-Khwarizmi αρχικά ως «*algorismus*». Ο τελευταίος, περίπου το 825 μ.Χ. εκπόνησε μια διατριβή στην αραβική γλώσσα, η οποία περιείχε συστηματικές τυποποιημένες λύσεις αλγεβρικών προβλημάτων και αποτελεί ίσως την πρώτη πλήρη πραγματεία άλγεβρας. Κατά τον Μεσαίωνα, algorismus σήμαινε το δεκαδικό σύστημα αριθμών καθώς υπήρχαν αυτοί που έκαναν υπολογισμούς χρησιμοποιώντας τον άβακα (abacus) κι αυτοί που έκαναν χρήση του δεκαδικού αριθμητικού συστήματος (algorismus). Στο λεξικό Webster’s New World Dictionary αναφέρθηκε για πρώτη φορά το 1957 το λήμμα «algorism» με τη σημασία της διαδικασίας του να κάνεις αριθμητική χρησιμοποιώντας το αραβικό αριθμητικό σύστημα.

Μέχρι την Αναγέννηση, η ετυμολογία της λέξης δεν ήταν πλήρως γνωστή.

Σταδιακά, η μορφή αλλά και η σημασία της λέξης «*algorismus*» αλλοιώθηκαν. Κατά τον 15ο αι., επηρεασμένη από την ελληνική λέξη «αριθμητικός», εμφανίστηκε στη λατινική γλώσσα ο όρος «*algorithmus*». Όπως αναφέρεται και στο Oxford English Dictionary (2012), η λέξη «*πέρασε από πολλές ψευτο-ετυμολογικές αλλαγές, συμπεριλαμβανομένης της λέξης αλγόριθμος που προέκυψε από συνδυασμό με την ελληνική λέξη αριθμητικός*».

Τη σύγχρονη σημασία του, κυρίως στον κλάδο των Μαθηματικών και της Επιστήμης των Υπολογιστών, ο όρος την οφείλει στη γρήγορη ανάπτυξη των ηλεκτρονικών υπολογιστών στα μέσα του 20ου αι.. Αν και δεν υπάρχει επίσημος ορισμός, μερικές κοινώς αποδεκτές θέσεις στις οποίες εδράζεται η έννοια του αλγορίθμου είναι οι εξής:

«*Ο αλγόριθμος ενός υπολογιστή είναι μια σειρά από βήματα για να ολοκληρωθεί μία εργασία η οποία περιγράφεται με τόση ακρίβεια ώστε να μπορεί να εκτελεστεί από ένα υπολογιστικό σύστημα*» (T. H. Cormen, 2013).

«*Ως αλγόριθμος μπορεί να θεωρηθεί ότι είναι οποιαδήποτε ακολουθία διαδικασιών μπορεί να προσομοιωθεί από ένα σύστημα που πληροί τις προϋποθέσεις Τούρινγκ (Turing Completeness)*» (Minsky 1967, Gurevich 2000).

## 2.3 Σχεδιασμός και ανάπτυξη αλγορίθμου

Ειρωνικά ως προς την έννοιά του, δεν υπάρχει κάποιο αυστηρώς καθορισμένο πρότυπο για την ανάπτυξη αλγορίθμων. Όσον αφορά στην Επιστήμη των Υπολογιστών, η ανάπτυξη ενός αλγορίθμου εξαρτάται από πολλούς παράγοντες αλλά κυρίως από τη φύση του προβλήματος που πρέπει να επιλυθεί, από τους διαθέσιμους υπολογιστικούς πόρους και από τον χρόνο εκτέλεσής του (π.χ. Big-O notation κ.α.).

Συνοπτικά, για να δημιουργηθεί ένας αλγόριθμος είναι ανάγκη να κατανοηθεί πλήρως το πρόβλημα και στη συνέχεια να οριστούν οι διαδικασίες (και η σειρά τους) που απαιτούνται για την επίλυσή του. Στο τέλος, είναι απαραίτητο να γίνει δοκιμή του αλγορίθμου.

Πιο αναλυτικά, τα τυπικά βήματα ανάπτυξης αλγορίθμων είναι τα εξής (Goodrich & Tamassia, 2002):

1. Ορισμός του προβλήματος
2. Ανάπτυξη ενός μοντέλου
3. Προσδιορισμός του αλγορίθμου
4. Σχεδιασμός ενός αλγόριθμου
5. Έλεγχος της ορθότητας του αλγορίθμου
6. Ανάλυση αλγορίθμου
7. Εφαρμογή αλγορίθμου
8. Δοκιμή προγράμματος
9. Προετοιμασία τεκμηρίωσης

Ένα πρόβλημα μπορεί, συνήθως, να επιλυθεί με πολλούς διαφορετικούς αλγορίθμους, όμως αυτό δεν σημαίνει απαραίτητα ότι είναι όλοι τους το ίδιο «αποδοτικοί». Υπάρχουν διαφορετικές μεθοδολογίες (paradigms) για τη δημιουργία αλγορίθμων.

Ανάλογα με τη φύση του προβλήματος, μια συγκεκριμένη μεθοδολογία μπορεί να είναι «πιο αποδοτική» από τις υπόλοιπες. Οι βασικές μέθοδοι δημιουργίας αλγορίθμων είναι οι εξής (Carroll & Daughtrey, 2007):

* Δημιουργία και δοκιμή ([Brute-force](https://en.wikipedia.org/wiki/Brute_force_search) ή exhaustive search): Αλγόριθμος που δοκιμάζει κάθε πιθανή λύση για να καταλήξει στην καλύτερη.
* Διαίρει και βασίλευε (Divide and conquer): Αλγόριθμος που  μειώνει την εμφάνιση ενός προβλήματος σε μία ή περισσότερες μικρότερες περιπτώσεις του ίδιου προβλήματος (συνήθως αναδρομικά) μέχρις ότου οι περιπτώσεις είναι αρκετά μικρές για να επιλυθούν εύκολα.
* Αναζήτηση και αρίθμηση (Search and enumeration): Αλγόριθμος που χρησιμοποιείται για δημιουργία κανόνων κίνησης σε ένα γραφικό περιβάλλον (π.χ. σκάκι).
* Τυχαίος αλγόριθμος (Randomized algorithm): Αλγόριθμος που κάνει επιλογές τυχαία.
* Μείωση πολυπλοκότητας ([Reduction of complexity](https://en.wikipedia.org/wiki/Reduction_(complexity))): Αλγόριθμος που μετασχιματίζει ένα δύσκολο πρόβλημα σε ένα πιο γνωστό πρόβλημα.
* Αναστροφή ([Back tracking](https://en.wikipedia.org/wiki/Back_tracking)): Αλγόριθμος που βρίσκει σταδιακά πολλαπλές λύσεις οι οποίες εγκαταλείπονται όταν διαπιστώνεται ότι δεν μπορούν να οδηγήσουν σε μια έγκυρη πλήρη λύση.

Δεν μπορούν όλες οι διαδικασίες επίλυσης ενός προβλήματος να θεωρηθούν «αλγόριθμοι». Οι αλγόριθμοι θα πρέπει να πληρούν κάποια πρότυπα και να διατυπώνονται με συγκεκριμένο τρόπο.

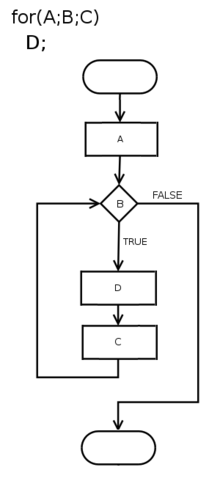
Έτσι, ένας αλγόριθμος πρέπει να ικανοποιεί τα επόμενα κριτήρια:

* Καθοριστικότητα (Unambiguous): Κάθε βήμα του είναι αυστηρά καθορισμένο ως προς το παραγόμενο αποτέλεσμα (Παναγιωτακόπουλος, 2013).
* Περατότητα (Finiteness): Κάθε εκτέλεση είναι πεπερασμένη, δηλαδή τελειώνει ύστερα από έναν πεπερασμένο αριθμό διεργασιών ή βημάτων (Παναγιωτακόπουλος, 2013).
* Αποτελεσματικότητα (Feasibility): Είναι μηχανιστικά αποτελεσματικός, δηλαδή λειτουργεί δίνοντας σωστά αποτελέσματα σε πεπερασμένη χρονική διάρκεια (Παναγιωτακόπουλος, 2013).
* Ανεξαρτησία (Independence): Ένας αλγόριθμος αποτελείται από βήμα προς βήμα οδηγίες που δεν εξαρτώνται από καμία γλώσσα προγραμματισμού.
* Να έχει είσοδο δεδομένων, επεξεργασία και έξοδο αποτελεσμάτων (input and output): Κατά την εκκίνηση εκτέλεσης του αλγορίθμου καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο, ενώ δίδει τουλάχιστον ένα μέγεθος ως αποτέλεσμα που εξαρτάται κατά κάποιο τρόπο από τις αρχικές εισόδους.

## 2.4 Δημιουργία και αναπαράσταση αλγορίθμου

Υπάρχουν τέσσερις βασικοί τρόποι αναπαράστασης ενός αλγορίθμου (Παναγιωτακόπουλος, 2013):

1. Ελεύθερο κείμενο, που αποτελεί τον πιο αδόμητο τρόπο παρουσίασης αλγορίθμου.
2. «*Δομημένη ομιλούμενη γλώσσα*» που εκτελείται κατά βήματα.
3. «*Διάγραμμα ροής*» (flowchart), που συνιστά έναν πιο γραφικό τρόπο παρουσίασης του αλγορίθμου, όπως φαίνεται παρακάτω:



Κύκλοι: Σύμβολα έναρξης-λήξης

Βέλη: Ροή/πορεία ελέγχου

Ρόμβοι: Υποθέσεις/Αποφάσεις

Ορθογώνια παραλληλόγραμα: Στάδια επεξεργασίας

Εικόνα . Ένα απλό διάγραμμα ροής

1. «Ψευδοκώδικας»: Κωδικοποίηση του αλγορίθμου σε ψευδογλώσσα.
2. Χρήση σύγχρονων γλωσσών προγραμματισμού

|  |  |
| --- | --- |
| Ψευδοκώδικας | Γλώσσα προγραμματισμού Python |
| **Αλγόριθμος** Παράδειγμα\_1  Sum ← 0  **Για i από 1 μέχρι 100**  Sum ← Sum + 1  **Τέλος\_επανάληψης**  **Εκτύπωσε** Sum  **Tέλος** Παράδειγμα\_1 | Sum = 0  **for** i in **range** (1,101):  Sum = Sum+1  **print** Sum |

Πίνακας 1. Παράδειγμα αλγορίθμου που εμφανίζει τους αριθμούς από το 1 έως το 100

Στον προγραμματισμό χρησιμοποιείται περισσότερο το διάγραμμα ροής, η ψευδογλώσσα και, φυσικά, η γλώσσα προγραμματισμού.

## 2.5 Αλγόριθμος και δημιουργικότητα

*«Clearly, I reject the view that there is one way*

*that is right for everyone and for every problem.»*

*-Bjarne Stroustrup*

Όπως αναφέρθηκε προηγουμένως, αλγόριθμος είναι μία μέθοδος επίλυσης προβλημάτων, η οποία αποτελείται από σαφώς καθορισμένες οδηγίες. Ο «αλγοριθμικός τρόπος σκέψης» είναι ένας όρος που χρησιμοποιείται συχνά ως μια από τις σημαντικότερες ικανότητες που μπορούν να επιτευχθούν μέσω της εκπαίδευσης των ατόμων σε θέματα Πληροφορικής και Επιστήμης των Υπολογιστών.

Ο αλγοριθμικός τρόπος σκέψη αφορά -κατά κάποιο τρόπο- σε ένα εύρος δεξιοτήτων που σχετίζονται με τη δημιουργία και την κατανόηση των αλγορίθμων: Η ικανότητα ανάλυσης ενός συγκεκριμένου προβλήματος, η ικανότητα ακριβούς καθορισμού του προβλήματος, η ικανότητα αναγνώρισης και δημιουργίας των απαραίτητων διαδικασιών για την επίλυση του προβλήματος, η ικανότητα αναγνώρισης όλων των πιθανών περιπτώσεων που αφορούν στο πρόβλημα, η ικανότητα βελτιστοποίησης της αποδοτικότητας (υπολογιστικοί πόροι, χρόνος διεκπεραίωσης κ.α.) του αλγορίθμου (Futschek,2006).

Με βάση τα παραπάνω, φαίνεται πως ο αλγοριθμικός τρόπος σκέψης έχει μια δημιουργική πτυχή: τη δημιουργία νέων αλγορίθμων που λύνουν ένα συγκεκριμένο πρόβλημα.

Δεν υπάρχει μια σαφής απάντηση στο ερώτημα «Πώς διδάσκεται η δημιουργικότητα;». Μια πρακτική απάντηση μπορεί να είναι: «Με την εξάσκηση στην επίλυση προβλημάτων» (Futschek,2006). Ειδικά για τους αρχάριους στην ανάπτυξη αλγορίθμων, τα προβλήματα πρέπει να μπορούν να επιλυθούν από οποιαδήποτε γλώσσα προγραμματισμού και να είναι όσο το δυνατόν ανεξάρτητα από τεχνικές λεπτομέρειες της κάθε γλώσσας (π.χ. να έχουν αυτόματη διαχείριση μνήμης), έτσι ώστε οι μαθητές να μπορούν να επικεντρωθούν περισσότερο στον σχεδιασμό του αλγορίθμου.

Συνεπώς, είναι προτιμότερο για τη δημιουργία ενός αλγορίθμου, η γλώσσα προγραμματισμού να είναι υψηλού ή πολύ υψηλού επιπέδου και γενικής χρήσης (π.χ. Python). Όσον αφορά στο πρόβλημα, η λύση του δε χρειάζεται να είναι πολύ απλή αλλά η εκφώνηση είναι απαραίτητο να είναι εύληπτη. Τα πιο σύνθετα προβλήματα επιτρέπουν και πιο δημιουργικές και ασυνήθιστες λύσεις από τους μαθητές.

# **Προγραμματισμός**

*«Και είπε τότε ο Κύριος:*

*ιδού έως τώρα ένας λαός είναι αυτοί και μίαν γλώσσαν ομιλούν όλοι»*

*-Γένεσις, 11,6*

## Τι είναι προγραμματισμός

Προγραμματισμός είναι η διαδικασία σχεδιασμού και δημιουργίας ενός εκτελέσιμου υπολογιστικού προγράμματος για την αυτοματοποιημένη εκτέλεση εργασιών ή επίλυση κάποιου υπολογιστικού προβλήματος από έναν Ηλεκτρονικό Υπολογιστή (Aράπογλου κ.α., 2006). Περιλαμβάνει εργασίες όπως: ανάλυση προβλήματος, δημιουργία αλγορίθμου, έλεγχο ακρίβειας αλγορίθμων και κατανάλωσης πόρων και εφαρμογή αλγορίθμου με τη χρήση γλώσσας προγραμματισμού.

Το πρόγραμμα εκτελείται από την Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ) του Ηλεκτρονικού Υπολογιστή, τον επεξεργαστή, αφού πρώτα ο πηγαίος κώδικας (οι εντολές που έχει γράψει ο προγραμματιστής σε μια γλώσσα προγραμματισμού) μεταφραστεί σε γλώσσα μηχανής από εξειδικευμένο λογισμικό, τους μεταγλωττιστές (compilers) ή/και τους διερμηνείς (interpreters). Θεωρητικά, ο πηγαίος κώδικας από κάθε γλώσσα προγραμματισμού μπορεί να μεταφραστεί σε γλώσσα μηχανής, η οποία είναι η μόνη γλώσσα που «καταλαβαίνει» ο επεξεργαστής.

Όπως συμβαίνει και με τον αλγόριθμο, ένα πρόγραμμα για να θεωρείται ορθώς σχεδιασμένο πρέπει να ικανοποιεί κάποια κριτήρια. Τα πιο σημαντικά από αυτά είναι ([Jones](https://en.wikipedia.org/wiki/Capers_Jones) & Bonsignour, 2011):

* Αξιοπιστία (Reliability): Πόσο συχνά το πρόγραμμα επιστρέφει σωστά αποτελέσματα, υπό την έννοια της σωστής σχεδίασης των αλγορίθμων και όχι των τεχνικών λεπτομερειών της γλώσσας προγραμματισμού (π.χ. διαίρεση με το 0).
* Ευρωστία (Robustness): Πώς το πρόγραμμα διαχειρίζεται σφάλματα, π.χ. εισαγωγή λανθασμένων δεδομένων από το χρήστη, διαχείριση περιορισμένης μνήμης RAM, ξαφνική διακοπή σύνδεσης με το Internet ή άλλη συσκευή ή διακοπή ρεύματος κ.α.
* Χρηστικότητα (Usability): Η ευκολία με την οποία ο χρήστης μπορεί να χρησιμοποιήσει το πρόγραμμα για τον σκοπό για τον οποίο έχει δημιουργηθεί (π.χ. εύκολο και διαισθητικό γραφικό περιβάλλον).
* Φορητότητα (Portability): Το εύρος των επεξεργαστών και λειτουργικών συστημάτων που μπορούν να εκτελέσουν σωστά το πρόγραμμα.
* Διατηρησιμότητα (Maintainability): Η ευκολία με την οποία το πρόγραμμα μπορεί να παραμετροποιηθεί με σκοπό τη βελτίωση της αποδοτικότητας ή/και των λειτουργιών του.
* Αποδοτικότητα (Efficiency): Η αποδοτικότερη διαχείριση των υπολογιστικών πόρων του συστήματος στο οποίο εκτελείται το πρόγραμμα.

## 3.1.1 Ιστορική αναδρομή

Η ιστορία εξέλιξης των γλωσσών προγραμματισμού ξεκινά σχεδόν μαζί με την ιστορία εξέλιξης των Ηλεκτρονικών Υπολογιστών (Παναγιωτακόπουλος, 2013).

Οι πρώτοι Ηλεκτρονικοί Υπολογιστές, όπως ο Κολοσσός, προγραμματίζονταν χωρίς τη βοήθεια προγραμμάτων αποθηκευμένων στη μνήμη του υπολογιστή, τροποποιώντας τα κυκλώματά τους (modifying their circuitry) ή τοποθετώντας γέφυρες φυσικών ελέγχων (banks of physical controls).

Αργότερα, κατά τη χρονική περίοδο από 1940 έως το 1953, αν και η έννοια του λειτουργικού συστήματος δεν υφίσταται, ήταν δυνατόν να γράφονται προγράμματα σε Γλώσσα Μηχανής (Machine Language), με την οποία οι προγραμματιστές χρησιμοποιούσαν ένα σύστημα οδηγιών σε μορφή ακολουθιών δυαδικών ψηφίων (bit) άμεσα εκτελέσιμες από την Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ) του υπολογιστή. Τα προγράμματα διαβάζονταν από τον Η/Υ μέσω διάτρητων καρτών, χάρτινων ταινιών, μαγνητικών ταινιών ή μέσω εναλλαγής διακοπτών στην πρόσοψη του υπολογιστή. Αυτού του είδους οι γλώσσες αργότερα ορίστηκαν ως γλώσσες προγραμματισμού πρώτης γενιάς.

Η δεύτερη γενιά γλωσσών προγραμματισμού ταυτίζεται με τις συμβολικές γλώσσες (Assembly Languages). Οι γλώσσες αυτές έχουν τις ίδιες δομές και εντολές με μία γλώσσα μηχανής, αλλά δίνουν τη δυνατότητα σε έναν προγραμματιστή να χρησιμοποιεί ονόματα αντί αριθμών. Το πρόγραμμα καταγραφόταν, δηλαδή, αρχικά με ειδικούς σύντομους συμβολισμούς ή συντομογραφίες, ακολουθούσε μετάφραση σε Γλώσσα Μηχανής μέσω του Συμβολομεταφραστή (Assembler) και εισαγωγή δεδομένων στον υπολογιστή (Παναγιωτακόπουλος, 2013).

Οι πρώτες γλώσσες προγραμματισμού υψηλού επιπέδου (High-level Languages), οι γλώσσες προγραμματισμού τρίτης γενιάς, δημιουργήθηκαν κατά τη δεκαετία του 1950. Πρόκειται για γλώσσες που χρησιμοποιούν εντολές άμεσα κατανοήσιμες από τον άνθρωπο, αφού συνήθως είναι λέξεις της καθομιλούμενης αγγλικής (Παναγιωτακόπουλος, 2013). Δημιουργήθηκαν για εξαλείψουν των γλωσσών προγραμματισμού της πρώτης και της δεύτερης γενιάς δίνοντας τη δυνατότητα στους προγραμματιστές να επικεντρωθούν στη λογική των προγραμμάτων χωρίς να χρειάζεται να λαμβάνουν υπόψη την εσωτερική αρχιτεκτονική των Η/Υ. Οι πρώτες σημαντικότερες γλώσσες υψηλού επιπέδου ήταν η FORTRAN (FORmula TRANslator – Μεταφραστής Τύπων), η COBOL (Common Business Oriented Language – Γλώσσα Προσανατολισμένη σε Κοινές Εμπορικές Εφαρμογές), η ALGOL (ALGOrithmic Language – Αλγοριθμική Γλώσσα), η BASIC και η C (Παναγιωτακόπουλος, 2013).

Κατά τις δεκαετίες 1970 και 1980 άρχισαν να αναπτύσσονται οι γλώσσες προγραμματισμού τέταρτης και πέμπτης γενιάς, οι οποίες θεωρούνται γλώσσες πολύ υψηλού επιπέδου (Very High-level Languages) και γλώσσες τεχνητής νοημοσύνης (Artificial Intelligence Languages) αντίστοιχα.

Σήμερα, στο χώρο της Επιστήμης των Υπολογιστών χρησιμοποιούνται κυρίως η Python, η C++, η Java, η C, η C#, η PHP, η R, η JavaScript, η Go και η Assembly (σύμφωνα με στοιχεία από το Institute of Electrical and Electronics Engineers – IEEE <https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>).

## 3.1.2 Γλώσσες προγραμματισμού

*«A programming language is like a natural, human language*

*in that it favors certain metaphors, images, and ways of thinking.»*

*-Seymour Papert*

Μία γλώσσα προγραμματισμού είναι ένα τυποποιημένο σύνολο εντολών και οδηγιών που μπορούν να εκτελεστούν από τον επεξεργαστή, με δική της σύνταξη και σημασιολογία. Ένα τέτοιο τυποποιημένο σύστημα εντολών για να θεωρείται «γλώσσα προγραμματισμού» πρέπει να ικανοποιεί κάποια συγκεκριμένα κριτήρια, τα κριτήρια Turing (Turing completeness) (Herken, 1995).

Ανάλογα με το επίπεδο αφαιρετικότητας, το προγραμματιστικό υπόδειγμα και την κύρια χρήση για την οποία έχει σχεδιαστεί μια γλώσσα προγραμματισμού, μπορεί να κατηγοριοποιηθεί αντίστοιχα σε:

1. Γλώσσα υψηλού (high-level) ή χαμηλού επιπέδου (low-level): Μια γλώσσα προγραμματισμού θεωρείται υψηλού επιπέδου όταν πλησιάζει την ανθρώπινη γλώσσα και χαμηλού επιπέδου όταν είναι πιο κοντά στη γλώσσα μηχανής (Assembly). Οι γλώσσες υψηλού επιπέδου είναι πιο φιλικές προς τον προγραμματιστή, υστερούν, όμως, ως προς την ταχύτητα εκτέλεσης του προγράμματος και δεν επιτρέπουν την άμεση πρόσβαση στο «υλικό» (hardware) του συστήματος, καθώς οι εντολές τους δεν εκτελούνται «απευθείας» από τον επεξεργαστή.
2. Προστακτικές (imperative), διαδικαστικές (procedural), αντικειμενοστραφείς (object-oriented) κ.α. (Παναγιωτακόπουλος, 2013): Στο προστακτικό προγραμματιστικό υπόδειγμα ακολουθείται η προσέγγιση επάνω-κάτω (top-down), δηλαδή το πρόγραμμα είναι μια λίστα εντολών που εκτελούνται με τη σειρά. Το διαδικαστικό υπόδειγμα ακολουθεί την ίδια προσέγγιση με το προστακτικό, με τη διαφορά πως περιέχει «ρουτίνες» και «υπορουτίνες». Αντίθετα, στο αντικειμενοστραφές υπόδειγμα όλα τα δεδομένα και οι μέθοδοι (διαδικασίες) είναι «αντικείμενα» μιας «κλάσης». Η κλάση θεωρείται το προσχέδιο (blueprint) όλων των αντικειμένων που ανήκουν σε αυτήν. Μια κλάση είναι ένα πρότυπο που ορίζει τις κοινές μεταβλητές και μεθόδους (διαδικασίες) όλων των ομοειδών αντικειμένων. Κάθε αντικείμενο είναι «στιγμιότυπο» μιας κλάσης.
3. Γενικής χρήσης (general purpose) ή συγκεκριμένου πεδίου εφαρμογής (domain-specific): Μια γλώσσα γενικής χρήσης είναι σχεδιασμένη για να δημιουργείται λογισμικό για ένα ευρύ πεδίο εφαρμογών σε αντίθεση με γλώσσες προγραμματισμού που έχουν σχεδιαστεί για συγκεκριμένο σκοπό (π.χ. δημιουργία «προγραμμάτων οδήγησης» συσκευών).

## 3.2 Η γλώσσα προγραμματισμού Python

*«Speak English! said the Eaglet. I don’t know the meaning of half of*

*those long words, and I don’t believe you do either.»*

*-Lewis Carroll, Alice in Wonderland*

Η γλώσσα προγραμματισμού που χρησιμοποιείται για τις ανάγκες της παρούσας εργασίας είναι η Python.

Η Python είναι μια γλώσσα γενικού σκοπού (general-purpose), σχεδιασμένη, δηλαδή, για να χρησιμοποιείται στην επίλυση μεγάλου εύρους προβλημάτων και σε διάφορα πεδία του προγραμματισμού. Θεωρείται γλώσσα υψηλού επιπέδου (high-level), καθώς χρησιμοποιεί λεξιλόγιο που μοιάζει με -περιορισμένη- φυσική γλώσσα, και η εκτέλεση του κώδικά της γίνεται μέσω ενός διερμηνέα (interpreter), δηλαδή μέσω ενός λογισμικού που εκτελεί τον πηγαίο κώδικα απευθείας, χωρίς να χρειάζεται πρώτα να το μεταγλωττίσει (compiler) σε γλώσσα μηχανής. Είναι δυναμική γλώσσα (dynamically typed), χαρακτηριστικό που κάνει την προγραμματιστική εμπειρία πιο εύκολη καθώς δεν είναι υποχρεωτικό να ορίζονται κάθε φορά οι τύποι μεταβλητών. Διαχειρίζεται, τέλος, τη μνήμη του υπολογιστή με τέτοιο τρόπο ώστε αντικείμενα του προγράμματος (π.χ. μεταβλητές) που δε χρειάζονται από ένα σημείο και έπειτα να καταργούνται αυτόματα (garbage collection). Μέσω της Python ένα πρόβλημα μπορεί να προσεγγιστεί με διαφορετικές μεθόδους (paradigms): την αντικειμενοστραφή (object-oriented) ή την διαδικαστική μέθοδο προγραμματισμού (procedural) (Kuhlman, 2009), την οποία και θα ακολουθήσουμε.

## 3.2.1 Ιστορική αναδρομή της Python

Την Python εμπνεύστηκε ο Ολλανδός προγραμματιστής Guido van Rossum, όταν το Δεκέμβριο του 1989 αποφάσισε να δημιουργήσει έναν διερμηνέα για μία γλώσσα που θα σκεφτόταν (Tollervey, 2015). Ονόμασε το ερευνητικό του σχέδιο «Python», από το όνομα του διάσημου βρετανικού κωμικού θιάσου «Monty Python’s Flying Circus» (Tollervey, 2015). Ο ίδιος στα μέσα της δεκαετίας του 1980 είχε συμμετάσχει στη δημιουργία της εκπαιδευτικής γλώσσας προγραμματισμού «ABC», η οποία στόχευε σε μη επαγγελματίες προγραμματιστές και από την οποία και επηρεάστηκε για τη δημιουργία της Python (Tollervey, 2015).

Η πρώτη έκδοση της Python, που έδινε έμφαση στην αναγνωσιμότητα του κώδικα μέσω της επιβολής εσοχών, δημοσιεύτηκε το 1991. Υπήρξαν πολλές μετέπειτα εκδόσεις μέχρι τη δημοσίευση της Python 3.0 τον Δεκέμβριο του 2008. Η τελευταία αυτή έκδοση δεν ήταν συμβατή με τις προηγούμενες, γι’ αυτό και πολλά χαρακτηριστικά της ενσωματώθηκαν σε αυτές (backported), δημιουργώντας τις εκδόσεις 2.6.x και 2.7.x, ώστε τελικά να είναι δυνατή η μετάφραση του κώδικα της Python 2 σε κώδικα Python 3. Η γλώσσα ανανεώνεται συνεχώς, με τελευταία έκδοση την Python 3.7.3 τον Μάρτιο του 2019 (<https://www.python.org/downloads/>).

Ο Guido van Rossum συνέχισε, μαζί με χιλιάδες άλλους προγραμματιστές, να συνεισφέρει στη βελτίωση της γλώσσας μέχρι και τον Ιούλιο του 2018, που του δόθηκε από την κοινότητα της Python ο τιμητικός τίτλος «Benevolent dictator for life» (Tollervey, 2015).

## 3.2.2 Η Python στην Εκπαίδευση

*«Now, it’s my belief that Python is a lot easier than to teach to students programming and teach them C or C++ or Java at the same time because all the details of the languages are so much harder. Other scripting languages really don’t work very well there either.»*

*-Guido van Rossum*

Η επιλογή της γλώσσας προγραμματισμού με την οποία οι μαθητές και οι φοιτητές θα έχουν την πρώτη τους επαφή με τον προγραμματισμό καθορίζει σε μεγάλο βαθμό τις πρώτες αναπαραστάσεις που θα δημιουργήσουν οι μαθητές για τα προγραμματιστικά αντικείμενα και τις αλγοριθμικές δομές, όπως επίσης και την στάση των μαθητών απέναντι στον προγραμματισμό (Kelleher & Pausch, 2005; McIver, 1996).

Η Python είναι μια σύγχρονη γλώσσα προγραμματισμού που έχει κυριαρχήσει τα τελευταία χρόνια στην τριτοβάθμια εκπαίδευση διεθνώς αλλά και στη χώρα μας. Η απλότητα της γλώσσας και η μεγάλη ποικιλία βιβλιοθηκών που παρέχει είναι ο λόγος που χρησιμοποιείται όχι μόνο από τμήματα πληροφορικής αλλά και από τμήματα Μαθηματικών και Οικονομικών. Τα τελευταία δυο χρόνια η Python έχει εισαχθεί και στην επαγγελματική εκπαίδευση με τα μαθήματα Αρχές Προγραμματισμού (Αράπογλου, Βραχνός, Κανίδης κ.α. 2015) και Προγραμματισμός Υπολογιστών (Αράπογλου, Βραχνός, Κανίδης κ.α. 2016) της Β’ και Γ΄ ΕΠΑΛ του τομέα πληροφορικής. Η Python φαίνεται σαν μια εξαιρετική επιλογή για την εκπαίδευση και έχει υιοθετηθεί, επίσης, από πολλά πανεπιστήμια κυρίως στην Αμερική (Agarwal, et. al., 2008, Goldwasser, & Letscher, 2008, Shein, 2015, Guo, 2014).

Πιο συγκεκριμένα, χαρακτηριστικά της γλώσσας Python που την καθιστούν καταλληλότερη από άλλες γλώσσες προγραμματισμού για τη χρήση της στην εκπαίδευση είναι τα εξής:

* Γίνεται εύκολα κατανοητή (Tollervey 2015, Briggs 2013)

H Python ελαχιστοποιεί τον συντακτικό θόρυβο, λόγω της πολύ απλής και λιτής της σύνταξης η οποία σε κάποιες περιπτώσεις θυμίζει ψευδοκώδικα (Agarwal, et. al., 2008; Goldwasser, & Letscher, 2008; Shein, 2015). Χρησιμοποιεί λεξιλόγιο πολύ κοντά στην ανθρώπινη γλώσσα (Tollervey, 2015), χαρακτηριστικό που την καθιστά γλώσσα υψηλού επιπέδου, ενώ, σε αντίθεση με άλλες γλώσσες προγραμματισμού, όπως η Java, η JavaScript και η C, έχει απλή σύνταξη, χωρίς περιττά σύμβολα, όπως curly braces ( { } ) και semicolons ( ; ). Ταυτόχρονα, η αυστηρή δομή της γλώσσας επιβάλλει τις εσοχές (indentations), που καθορίζουν το εύρος μιας εργασίας, και δημιουργείται, έτσι, ένας καθαρός κώδικας αποτελούμενος από πολλά κομμάτια κώδικα (code blocks). Λαμβάνοντας υπόψη τα παραπάνω, η Python είναι μία γλώσσα προσβάσιμη τόσο για τη δημιουργία κώδικα όσο και για την κατανόηση του κώδικά της. Ταυτόχρονα τα μηνύματα λάθους είναι σαφή και λεπτομερή.

* Είναι γλώσσα ανοικτού κώδικα

Η Python είναι δωρεάν διατιθέμενη (free) στο διαδίκτυο και ανοικτού κώδικα (open-source). Είναι, δηλαδή, διαθέσιμη στο κοινό για να την προμηθευτεί, να την χρησιμοποιήσει, να την παραμετροποιήσει και να την διανείμει (FLOSS: Free/Libre and Open Source Software) χωρίς να παραβιάζει κάποια άδεια χρήσης λογισμικού . Έτσι, υπάρχει μία οργανωμένη ενεργή κοινότητα προγραμματιστών, η οποία έχει δημιουργήσει, ως σημείο αναφοράς, την Python Software Foundation (PSF, <https://www.python.org/psf/>) με σκοπό «την προώθηση, προστασία και εξέλιξη της γλώσσας προγραμματισμού Python, αλλά και την υποστήριξη και διευκόλυνση της ανάπτυξης μιας ποικιλόμορφης και διεθνούς κοινότητας προγραμματιστών της Python». Η Python έχει πολύ μεγάλη κοινότητα εκπαιδευτικών και προγραμματιστών, οι οποίοι διαθέτουν ελεύθερα υλικό (βιβλία, online μαθήματα, φύλλα εργασίας, tutorials, σημειώσεις) στο διαδίκτυο.

* Φορητότητα

Η Python λειτουργεί σε Microsoft Windows, Mac OS X, Linux καθώς και σε άλλα λειτουργικά συστήματα και συσκευές (Tollervey, 2015).

* Μεγάλη πρότυπη βιβλιοθήκη

Βασικό πλεονέκτημα της Python είναι η βιβλιοθήκη της. Πρόκειται για έτοιμες λειτουργίες, ενσωματωμένες μέσα στη γλώσσα, τις οποίες μπορούμε να επιλέξουμε και να εισάγουμε στον κώδικα χωρίς επιπλέον γραμμές κώδικα.

* Προοπτικές εξέλιξης

Σε αντίθεση με άλλες γλώσσες που προτείνονται για τη διδασκαλία αρχάριων στον προγραμματισμό (Logo, Scratch κ.α.), η Python δεν είναι μία γλώσσα προγραμματισμού που μπορεί να αξιοποιηθεί μόνο στο πλαίσιο της εκπαίδευσης (Guido van Rossum, 1999). Σύμφωνα με το Ινστιτούτο Ηλεκτρολόγων και Ηλεκτρικών Μηχανικών (Institute of Electrical and Electronics Engineers - IEEE), είναι η πρώτη πιο δημοφιλής γλώσσα που χρησιμοποιούν οι επαγγελματίες προγραμματιστές για το 2018 (<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>). Χρησιμοποιείται για την ανάπτυξη εφαρμογών με γραφικό περιβάλλον, δυναμικών ιστοσελίδων και διαδικτυακών εφαρμογών, ενώ είναι δημοφιλής και στο χώρο της τεχνητής νοημοσύνης (Artificial Intelligence - AI).

# **Η επιστήμη των Υπολογιστών στην εκπαίδευση**

## 4.1 Μοντέλα εισαγωγής της Επιστήμης των Υπολογιστών στην Εκπαίδευση

Τα τελευταία χρόνια υπάρχει έντονο ενδιαφέρον στη μελέτη σχετικά με το ρόλο και τα πιθανά οφέλη του προγραμματισμού και γενικότερα της χρήσης Η/Υ για όλους. Ως αποτέλεσμα αυτού, όλο και περισσότερες χώρες εντάσσουν δραστηριότητες ή μαθήματα που έχουν να κάνουν με τον υπολογιστή στο Αναλυτικό Πρόγραμμα Σπουδών (ΑΠΣ) του σχολείου (Heintz et al. 2016).

Η εισαγωγή νέου περιεχομένου στο ΑΠΣ εγείρει πάντα ερωτήματα όπως που να ενταχθεί, ποιο μάθημα να αντικαταστήσει και ποιος είναι κατάλληλος ή/και διαθέτει τα προσόντα να το διδάξει. Έτσι, ο τρόπος ένταξής της Επιστήμης των Υπολογιστών (Computer Science) ποικίλει από χώρα σε χώρα ανάλογα με την αντίληψη που υπάρχει σχετικά με το θέμα και τον διαθέσιμο χρόνο έτσι ώστε να ενταχθεί στο ΑΠΣ (Heintz et al. 2016).

Στην Αυστραλία, οι Ψηφιακές Τεχνολογίες (Digital Technologies - DT) διδάσκονται ως αυτοτελές αντικείμενο τόσο στην Α/θμια όσο και στην Β/μια εκπ/ση και επικεντρώνονται σε δεξιότητες υπολογιστικής σκέψης και στην ανάπτυξη τόσο ψηφιακού γραμματισμού όσο και κατανόηση των αρχών της υπολογιστικής σκέψης.

Ως ξεχωριστό αντικείμενο διδάσκεται και στην Αγγλία. Σε αυτή την περίπτωση το ενδιαφέρον επικεντρώνεται στον προγραμματισμό, ενώ περιλαμβάνονται στοιχεία Επιστήμης των Υπολογιστών, Τεχνολογίας των Πληροφοριών (Information Technology) και ψηφιακού γραμματισμού στοχεύοντας *1)* στην κατανόηση και εφαρμογή αρχών της Επιστήμης των Υπολογιστών, *2)* στην ανάλυση προβλημάτων με προγραμματιστικούς όρους και δημιουργία προγραμμάτων για την επίλυσή τους, *3)* στην αξιολόγηση και εφαρμογή Τεχνολογίας των Πληροφοριών για επίλυση προβλημάτων και *4)* στη συνειδητή χρήση της Τεχνολογίας των Πληροφοριών και της Επικοινωνίας.

Επίσης, στην Εσθονία το ProgeTiger programme υπάρχει στο ΑΠΣ για τη δημοτική εκπαίδευση με τρεις θεματικές ενότητες: *α)* engineering sciences, που αφορά στον προγραμματισμό και τη ρομποτική, *β)* Design and Technology, που αφορά στη 3D μοντελοποίηση, τα γραφικά, τα πολυμέσα και τη δημιουργία κινούμενων εικόνων (animation), *γ)* Τεχνολογία των Επικοινωνιών και της Πληροφορίας (ICT), που αφορά στη χρήση των υπολογιστών, των δικτύων και των τεχνολογικών εργαλείων.

Στη Νέα Ζηλανδία οι DT εισάγονται ως ξεχωριστό μάθημα στην Β/θμια εκπ/ση και περιλαμβάνουν τον προγραμματισμό και ένα μεγάλο εύρος θεμάτων Επιστήμης Υπολογιστών, χωρίς, ωστόσο, να διδάσκονται σε βάθος καθώς σκοπός είναι να αναπτυχθεί απλά μια γενική ιδέα σχετικά με την Επιστήμη των Υπολογιστών. Και στη Νορβηγία ο προγραμματισμός εισάγεται στη Β/θμια εκπ/ση, όπου είναι υποχρεωτική η μάθηση τουλάχιστον δύο γλωσσών προγραμματισμού, εκ των οποίων η μια γραπτή (text-based), σκοπό την αναγνώριση προβλημάτων, ανάπτυξη λύσεων, συστηματικός τρόπος εύρεσης σφαλμάτων και βελτίωση του κώδικα και καταγραφή με κατανοητό τρόπο το σκεπτικό της λύσης.

Στην Πολωνία η Επιστήμη των Υπολογιστών περιλαμβάνεται στην Α/θμια και Β/θμια εκπ/ση και αποτελείται από τρία στάδια: *1)* δραστηριότητες με τον Η/Υ, *2)* ICT, όπου οι Η/Υ χρησιμοποιούνται ως εργαλεία που υποστηρίζουν τη γνώση ποικίλων μαθημάτων, *3)* αλγόριθμοι, αλγοριθμική σκέψη και επίλυση προβλημάτων με τη χρήση Η/Υ. Ο προγραμματισμός εδώ δεν προτείνεται ρητά, αλλά τα περισσότερα σχολεία τον εισάγουν στο ΑΠΣ.

Στη Νότια Κορέα το «Informatics» αποτελεί υποχρεωτικό μάθημα στην Β/θμια εκπ/ση και καλύπτει θέματα ψηφιακού γραμματισμού, υπολογιστικής σκέψης και προγραμματισμού.

Στη Φινλανδία ο προγραμματισμός υπάρχει στην εκπαίδευση ως μέρος των μαθηματικών ενώ αποτελεί κομμάτι της ανάπτυξης ψηφιακών δεξιοτήτων και καλύπτει ένα ευρύ φάσμα αντικειμένων που εντάσσονται στην Επιστήμη των Υπολογιστών.

Ιδιόρρυθμη περίπτωση αποτελούν οι ΗΠΑ, αφού κάθε πολιτεία και σχολική κοινότητα είναι δυνατόν να έχουν το δικό τους ΑΠΣ. Ωστόσο, σε εθνικό επίπεδο προωθείται η εισαγωγή στην Επιστήμη των Υπολογιστών για όλους με σκοπό τη σφαιρική κατανόησή της. Προτείνεται η οργάνωσή της με βάση τις εφτά βασικές αρχές της Επιστήμης των Υπολογιστών: δημιουργικότητα, αφαιρετικότητα, δεδομένα και πληροφορία (data and information), αλγόριθμοι, προγραμματισμός, διαδίκτυο και παγκόσμιος αντίκτυπός του (Heintz et al. 2016).

Η γενική τάση είναι η εισαγωγή της Επιστήμης των Υπολογιστών, συχνά με τη μορφή της ανάπτυξης αρχικά της υπολογιστικής σκέψης στην Α/θμια εκπ/ση, ενώ στην Β/θμια εκπ/ση η τάση είναι να διατίθενται μαθήματα που προσφέρουν γνώσεις σε ένα ευρύτερο φάσμα της Επιστήμης των Υπολογιστών καθώς και της επίδρασής της στην κοινωνία. Πρόκληση αποτελεί για όλες τις χώρες είναι να εκπαιδεύσουν και να διατηρήσουν στο επάγγελμα τους εκπαιδευτικούς με τις σχετικές γνώσεις και δεξιότητες και να εντάξει ομαλά στο ΑΠΣ την Επιστήμη των Υπολογιστών (Heintz et al. 2016).

## 4.2 Η ένταξη του προγραμματισμού στην Εκπαίδευση: πλεονεκτήματα και προκλήσεις

Πληθώρα μελετών έχουν δείξει ότι η ένταξη μαθημάτων προγραμματισμού στην Α/θμια και Β/θμια εκπ/ση (K-12) μπορεί να είναι μια ευχάριστη, ενδιαφέρουσα και αποτελεσματική εμπειρία για τους μαθητές (Garneli et al. 2015).

Έχει αναδειχθεί μια μεγάλη ποικιλία εργαλείων και αντίστοιχων πρακτικών για την ενασχόληση των μαθητών με τον προγραμματισμό (Garneli et al. 2015, Hubweser et al. 2015).

Οι κειμενικές γλώσσες προγραμματισμού (textual programming languages) και επαγγελματικά εργαλεία προσφέρουν στους αρχάριους προγραμματιστές μια πιο αυθεντική εμπειρία ενασχόλησης με την Επιστήμη των Υπολογιστών, αποτελώντας έμπνευση και ταυτόχρονα πρόκληση για πολλούς μαθητές αλλά και για τους εκπαιδευτικούς. Τέτοιες γλώσσες θα μπορούσαν να προσελκύσουν το ενδιαφέρον μαθητών με υψηλότερες προσδοκίες και περισσότερες ανάγκες. Σε αυτή την περίπτωση οι μαθητές καλούνται να είτε να δημιουργήσουν το δικό τους κώδικα, με την καθοδήγηση και τη βοήθεια του εκπαιδευτικού ή των συνομηλίκων, είτε να μελετήσουν, επεξεργαστούν και επεκτείνουν έναν ήδη δοσμένο κώδικα. Είναι σημαντικό να αναφερθεί ότι μια τέτοια προσέγγιση μπορεί εύκολα να έχει εφαρμογή και επιτυχία στον περιορισμένο χρόνο μιας σχολικής τάξης (Garneli et al. 2015).

Οι οπτικές γλώσσες προγραμματισμού (visual programming languages) είναι κατάλληλες να χρησιμοποιηθούν ως εισαγωγικές γλώσσες για αρχάριους μαθητές. Εκτός του αλγοριθμικού τρόπου σκέψης, της ικανότητας διαχείρισης λαθών και τον ψηφιακό γραμματισμό που προσφέρουν όλες οι μορφές ενασχόλησης με τον προγραμματισμό, οι οπτικές γλώσσες είναι προσιτές για περισσότερους ανθρώπους ενώ περιλαμβάνουν χαρακτηριστικά που τις καθιστούν κατάλληλες για άτομα με αναπηρίες (Garneli et al. 2015).

Τα απτά περιβάλλοντα (tangible interface) προάγουν τη συνεργατική μάθηση και είναι ευκολότερα στη χρήση τους κυρίως για τα μικρότερα παιδιά. Τα παιδιά μεγαλύτερης ηλικίας δεν τα θεωρούν ευκολότερα, ωστόσο τα απολαμβάνουν. Μπορούν να υποστηρίξουν κονστρακσιονιστικές πρακτικές, όπως επίλυση προβλημάτων, έκφραση μέσω της κατασκευής απτών πραγμάτων κ.α.. Υποστηρίζονται από ποικίλες παρουσιάσεις/οδηγίες και διδακτικό υλικό που ακολουθείται από δημιουργικό μάθημα.

Η απόφαση σχετικά με το ποια από τις παραπάνω πρακτικές είναι η καταλληλότερη καθώς και με ποιον τρόπο πρέπει να προσεγγιστεί εξαρτάται από τους μαθησιακούς στόχους που τίθενται και από το γνωστικό υπόβαθρο, τις ανάγκες, τα ενδιαφέροντα και τις ικανότητες των εκάστοτε μαθητών (Garneli et al. 2015).

Μερικές προσεγγίσεις που εμφανίζονται είναι οι εξής:

* Problem-project based: οι μαθητές εφαρμόζουν τις γνώσεις τους λύνοντας προβλήματα ή παράγοντας έργο (project)
* Use-modify-create: σταδιακή ανάπτυξη παιχνιδιών
* «in-time» pedagogy: παρέχεται νέα γνώση όποτε είναι απαραίτητο, μέσω δραστηριοτήτων και με μία προσέγγιση βασισμένη σε παραγωγή έργου (project-based)
* Inquiry Based Learning: ο εκπαιδευτικός παρουσιάζει το πρόβλημα και μέσω ερωτήσεων βοηθάει τους μαθητές να το λύσουν

Είναι, πάντως, σημαντικό τα μαθήματα να γίνονται με τρόπο δημιουργικό, όπου οι μαθητές δημιουργούν/επεξεργάζονται ένα έργο ή λύνουν προβλήματα και, φυσικά, το περιβάλλον μάθησης να διέπεται από θετικές σχέσεις, συνεργατικότητα, υψηλούς στόχους και υγιή ψυχολογική προσαρμογή (Garneli et al. 2015).

# **Σκοπός της εργασίας**

Σκοπός της παρούσας εργασίας είναι η ανάδειξη της χρησιμότητας της διδασκαλίας του προγραμματισμού ως γνωστικό εργαλείο για την ενίσχυση της αλγοριθμικής σκέψης των μαθητών και η δημιουργία κατάλληλου εκπαιδευτικού υλικού.

Η διεθνής βιβλιογραφία έχει αναδείξει πως η επιλογή της γλώσσας προγραμματισμού με την οποία οι μαθητές θα έχουν την πρώτη τους επαφή με τον προγραμματισμό καθορίζει σε μεγάλο βαθμό τις πρώτες αναπαραστάσεις που θα δημιουργήσουν για τις αλγοριθμικές δομές και την στάση τους απέναντι στον προγραμματισμό.

Για τον λόγο αυτό για την δημιουργία του εκπαιδευτικού υλικού επιλέχθηκε η γλώσσα προγραμματισμού Python, καθώς είναι μια ανοικτού κώδικα γλώσσα εύκολα κατανοητή για τον αρχάριο προγραμματιστή και με προοπτικές εξέλιξης.

Στόχος της εργασίας δεν είναι η εκμάθηση της γλώσσας προγραμματισμού αυτής καθ’ αυτής, αλλά η ανάπτυξη ή/και ενίσχυση των δεξιοτήτων επίλυσης προβλημάτων μέσω αυτής. Αυτό επιτυγχάνεται αφενός γιατί η έκφραση σε μια γλώσσα προγραμματισμού απαιτεί σαφή διατύπωση του αλγορίθμου επίλυσης και αφετέρου γιατί δεν υποδεικνύεται, όταν δεν είναι απαραίτητο, προς χάριν της δημιουργικότητας ένας συγκεκριμένος τρόπος επίλυσης.

Σε αυτό το πλαίσιο αναπτύχθηκε εκτενές εκπαιδευτικό υλικό για τον εκπαιδευτικό, σύντομες οδηγίες για την αξιοποίηση της Python στην τάξη και ενδεικτικά φύλλα εργασίας με δραστηριότητες για τους μαθητές. Κατά τη διάρκεια των δραστηριοτήτων οι μαθητές μαθαίνουν σταδιακά πως να προγραμματίζουν (λεξιλόγιο, σύνταξη, τεχνικές λεπτομέρειες κ.α.) και ταυτόχρονα να επιλύουν ένα πρόβλημα κάνοντας χρήση των όσων έμαθαν.

Η δημιουργία αυτού του εκπαιδευτικού υλικού αποσκοπεί στην υποστήριξη των εκπαιδευτικών, ειδικά μικρής ηλικίας μαθητών, στην διδασκαλία του προγραμματισμού ως γνωστικό εργαλείο.

# **Προτάσεις για περεταίρω έρευνα**

Σημαντικής σημασίας είναι ο σχεδιασμός και υλοποίηση εφαρμογής και αξιολόγησης του παραχθέντος υλικού:

* Με μαθητές Ε’ και Στ’ τάξης Δημοτικού Σχολείου
* Με μαθητές Γυμνασίου
* Με μελλοντικούς εκπαιδευτικούς Α/θμιας εκπ/σης
* Με εν ενεργεία εκπαιδευτικούς Α/θμιας εκπ/σης
* Με εν ενεργεία εκπαιδευτικούς Β/θμιας εκπ/σης

Η αξιολόγηση αυτή θα δώσει δεδομένα για την αξιοποίηση της γλώσσας προγραμματισμού Python κατά τη διδασκαλία του προγραμματισμού, ειδικά σε μικρής ηλικίας μαθητές, ως γνωστικού εργαλείου για την υποστήριξη της αλγοριθμικής σκέψης των μαθητών.

# 2o ΜΕΡΟΣ

# Εκπαιδευτικό υλικό για τον εκπαιδευτικό

***Λίγα λόγια για τη γλώσσα προγραμματισμού Python***

Το υλικό αυτό αφορά στην Python. Δημιουργήθηκε από τον Ολλανδό Guido van Rossum. Πρόκειται για μια γλώσσα με πολύ σαφή σύνταξη και δομή, αρκετά κοντά στην ανθρώπινη γλώσσα, άρα πολύ κατανοητή.

Η **Python** είναι μία γλώσσα προγραμματισμού που χρησιμοποιείται για την κατασκευή εφαρμογών στο διαδίκτυο αλλά και απλών και σύνθετων εφαρμογών για όλα τα λειτουργικά συστήματα (όπως Windows, Linux, Android κ.α.). Τα τελευταία χρόνια έχει καθιερωθεί ως η βασική γλώσσα προγραμματισμού στο χώρο της επιστήμης (data science, machine learning κ.α.). Είναι πολύ διαδεδομένη στον χώρο οργανισμών όπως η NASA, η Google, η CIA, η Disney και ιδιαίτερα στους τομείς της ρομποτικής και της τεχνητής νοημοσύνης (<https://wiki.python.org/moin/OrganizationsUsingPython>).

Για τη συγγραφή προγραμμάτων είναι απαραίτητος ένας κειμενογράφος ή ακόμα καλύτερα ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE), το οποίο είναι ένα ειδικό λογισμικό για την ανάπτυξη εφαρμογών. Η Python έρχεται μαζί με ένα εύχρηστο περιβάλλον ανάπτυξης με την ονομασία **ΙDLE**. Τα αρχικά του έρχονται από τις λέξεις Interactive DeveLopment Environment. Μερικά IDE ανάπτυξης κώδικα σε python είναι το Eric, το Atom, το PyCharm, το Visual Studio Code (VSCode) κ.α..

Εμείς θα χρησιμοποιήσουμε το τελευταίο, καθώς είναι το πιο κατάλληλο για μικρά προγράμματα, όπως αυτά με τα οποία θα ασχοληθούμε εμείς, ενώ καταλαμβάνει το λιγότερο χώρο στον υπολογιστή σε σχέση με τα υπόλοιπα. Το γραφικό περιβάλλον του VSCode είναι ιδιαίτερα φιλικό προς το χρήστη, συμπέρασμα στο οποίο μας οδηγούν τρία (3) βασικά χαρακτηριστικά του:

* Επισύμανση σύνταξης *(*[*syntax highlighting*](https://en.wikipedia.org/wiki/Syntax_highlighting)*)* με διαφορετικά χρώματα ανάλογα με την κατηγορία που ανήκει ο κάθε όρος (εντολή, λειτουργία, μεταβλητή κ.α.)
* Έξυπνη συμπλήρωση κώδικα *(*[*intelligent code completion*](https://en.wikipedia.org/wiki/Intelligent_code_completion)*)*, που επιταχύνει τη διαδικασία συγγραφής προγραμμάτων και μειώνει τη συχνότητα των τυπογραφικών λαθών
* Υποστήριξη αποσφαλμάτωσης *(debugging)*, διαδικασία που εντοπίζει και υποδυκνύει τυχόν ελλείψεις ή προβλήματα στον υπό πραγμάτωση κώδικα

Στα παρακάτω παραδείγματα θα χρησιμοποιήσουμε την python για να δούμε πως μπορούμε να λύσουμε με τη βοήθεια του υπολογιστή απλά αλλά και πιο σύνθετα προβλήματα υπολογισμών που συναντάμε στην καθημερινότητά μας.

***Βασικά στοιχεία της γλώσσας προγραμματισμού Python***

Εντολή print

Ας ξεκινήσουμε με τη δημιουργία ενός προγράμματος που εμφανίζει στην οθόνη του υπολογιστή μια φράση (π.χ. τη φράση «Hello world!»).

Για αυτό, στην Python χρησιμοποιούμε την εντολή **print**.

*Παράδειγμα σύνταξης:*

print('Hello world!')

Τελεστές και πράξεις

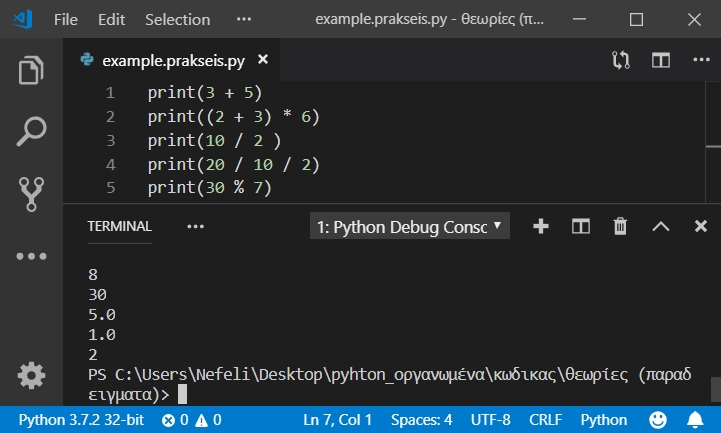
Τα παρακάτω σύμβολα πραγματοποιούν τις αντίστοιχες πράξεις όπως φαίνονται στον πίνακα:

|  |  |
| --- | --- |
| **+** | Πρόσθεση |
| **-** | Αφαίρεση |
| **\*** | Πολλαπλασιασμός |
| **\*\*** | Ύψωση σε δύναμη |
| **/** | Διαίρεση |
| **//** | Πηλίκο ακέραιας διαίρεσης (πόσες φορές χωράει ένας αριθμός σε έναν άλλον) |
| **%** | Υπόλοιπο ακέραιας διαίρεσης (το υπόλοιπο ενός αριθμού αν διαιρεθεί με έναν άλλον) |

*Παραδείγματα σύνταξης:*

print(3 + 5)

print((2 + 3) \* 6)



*Σημείωση: Στον Προγραμματισμό οι παρενθέσεις και η σειρά των πράξεων έχουν το ίδιο νόημα όπως και στα Μαθηματικά.*

Μεταβλητή - variable

Οι μεταβλητές είναι συμβολικά ονόματα που αντιστοιχούν σε θέσεις μνήμης του υπολογιστή. Σε αυτές τις θέσεις μνήμης αποθηκεύουμε διάφορες τιμές (π.χ. το σκορ ενός παιγνιδιού, τον αριθμό ζωών στο κτήμα μας, την απάντηση ενός χρήστη σε μια ερώτηση γνώσεων, κτλ). Οι τιμές αυτές μπορούν να αλλάξουν, να διαβαστούν, να αντιγραφούν ή να διαγραφούν.

Άρα, όταν δημιουργούμε μια μεταβλητή, ουσιαστικά δεσμεύουμε μια θέση στη μνήμη του υπολογιστή στην οποία μπορούμε να αποθηκεύσουμε όποια τιμή θελήσουμε.

*Παράδειγμα σύνταξης:*

x = 7

Με την ονομασία της μεταβλητής προσδιορίζουμε τον τρόπο με τον οποίο θα αναφερόμαστε στη συγκεκριμένη θέση μνήμης και θα αλλάζουμε τα περιεχόμενά της, δηλαδή την τιμή της. Καλό είναι στις μεταβλητές που δημιουργούμε να δίνουμε ονόματα με νόημα, δηλαδή ονόματα που προσδιορίζουν τη χρησιμότητα της μεταβλητής

Για να είναι έγκυρο το όνομα μιας μεταβλητής χρησιμοποιούνται χαρακτήρες όπως γράμματα, αριθμοί και κάτω παύλα ( \_ ).

*Παράδειγμα σύνταξης:*

ilikia\_mathiti = 7

a1 = 4

Περιορισμοί:

Το όνομα μιας μεταβλητής δημιουργεί σφάλμα αν ξεκινάει με αριθμούς ή περιέχει κενά. Επίσης, οι πεζοί χαρακτήρες είναι διαφορετικοί από τους κεφαλαίους.

*Παράδειγμα εσφαλμένης ονομασίας:*

ilikia mathiti = 7

1a = 4

Tύποι μεταβλητών: string , float , integer

Μερικοί βασικοί τύποι μεταβλητών είναι strings, floats, integers.

**συμβολοσειρά (string)**: είναι μια σειρά από γράμματα αριθμούς και άλλους χαρακτήρες και ορίζεται ανάμεσα σε ‘ ‘ ή “ “.

Παράδειγμα σύνταξης: name = “Νεφέλη”

Αριθμοί: **integers** (**int**), που είναι οι ακέραιοι αριθμοί, και σε **floats** (**float**), αριθμοί που δεν είναι ακέραιοι.

*Παράδειγμα σύνταξης:*

x = int(5)

x = float(5)

Το αποτέλεσμα μιας πράξης θα είναι αυτόματα **float**:

* Αν διαιρέσουμε δύο οποιουσδήποτε αριθμούς
* Αν κάνουμε μια πράξη με δύο δεκαδικούς αριθμούς
* Αν κάνουμε μια πράξη με έναν δεκαδικό και έναν ακέραιο αριθμό

Είσοδος δεδομένων - input

Τα προγράμματα υπολογιστών λαμβάνουν δεδομένα (input) και τα επεξεργάζονται για να παράγουν ένα αποτέλεσμα (output).

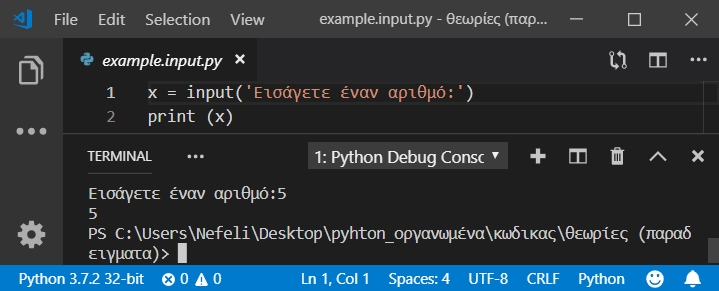
Η βασική λειτουργία που δείχνει στην οθόνη το αποτέλεσμα που παράγεται είναι η λειτουργία **print**.

Όταν θέλουμε να δώσουμε τη δυνατότητα στον χρήστη να εισάγει δεδομένα από το πληκτρολόγιο, χρησιμοποιούμε την εντολή **input**.

*Παράδειγμα σύνταξης:*

x = input('Εισάγετε έναν αριθμό:')

print (x)



Εντολή εάν - if

Χρησιμοποιούμε τη εντολή **εάν** (**if**) για να πραγματοποιηθεί μια εντολή στον κώδικα του προγράμματος μόνο αν μια συνθήκη είναι αληθής.

*Παράδειγμα σύνταξης:*

if 3 < 10:

print('Το 3 είναι μικρότερο του 10')

Μια εντολή **if** μπορεί να περιέχει μια άλλη συνθήκη **if** (εμφωλευμένο ή φωλιασμένο **if**). Αυτό σημαίνει πως η δεύτερη συνθήκη πραγματοποιείται μόνο αν ισχύει η πρώτη.

*Παράδειγμα σύνταξης:*

if 12 < 20:

print('Το 12 είναι μικρότερο του 20')

if 12 > 10:

    print('Το 12 είναι ανάμεσα στο 10 και το 20')

Η εντολή **else** ακολουθεί τη συνθήκη **if** και περιλαμβάνει κώδικα που πραγματοποιείται αν η συνθήκη αποδειχθεί ψευδής (**εάν…αλλιώς…**).

*Παράδειγμα σύνταξης:*

age = 15

if age >= 18:

print('Είσαι ενήλικος')

else:

print('Είσαι ανήλικος')

Η συνθήκη **elif** (συντομογραφία του **else if**) χρησιμοποιείται για μία σειρά συνθηκών

*Παράδειγμα σύνταξης:*

ilikia = int(input ('Γράψτε μία ηλικία:'))

if ilikia <= 12:

print('Πηγαίνει δημοτικό')

elif ilikia <= 15:

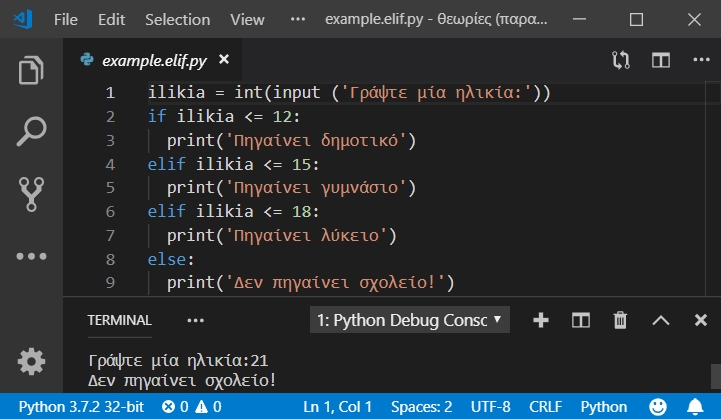
print('Πηγαίνει γυμνάσιο')

elif ilikia <= 18:

print('Πηγαίνει λύκειο')

else:

print('Δεν πηγαίνει σχολείο!')



Boolean & συγκρίσεις

**Boolean** είναι ένας τύπος μεταβλητής (δεδομένων) που μπορεί να πάρει μόνο μία από τις δύο τιμές **True** ή **False**.

*Ουσιαστικά, μπορεί να επιτευχθεί το ίδιο χρησιμοποιώντας μεταβλητές τύπου string ή integer, όμως για τεχνικούς λόγους, που δεν είναι στο πλαίσιο της παρούσας εργασίας, αυτός ο τύπος μεταβλητής είναι πολύ χρήσιμος και χρησιμοποιείται συχνά.*

Τιμή **True** ή **False** μπορεί να πάρει η σύγκριση (**comparison**) δύο τιμών :

* Με το σύμβολο: **==** , όπου η σύγκριση είναι ***αληθής (True)*** όταν τα δύο συγκρινόμενα αντικείμενα ταυτίζονται και ***ψευδής (False)*** όταν δεν ταυτίζονται.

*Παράδειγμα σύνταξης:*

3 == 5

False

'hello' == 'hello'

True

* Με το σύμβολο: !=, όπου η σύγκριση είναι ***αληθής (True)*** όταν τα δύο συγκρινόμενα αντικείμενα ΔΕΝ ταυτίζονται και ***ψευδής (False)*** όταν ταυτίζονται.

*Παράδειγμα σύνταξης:*

5 != 5

False

5 != 6

True

* Με το σύμβολο > ή < , όπου εξετάζεται αν ένας αριθμός είναι μεγαλύτερος ή μικρότερος από τον άλλον

*Παράδειγμα σύνταξης:*

5 < 4

False

6 > 5

True

* Με το σύμβολο >= ή <= , όπου εξετάζεται αν ένας αριθμός είναι μεγαλύτερος ή ίσος και μικρότερος ή ίσος, αντίστοιχα, από τον άλλον

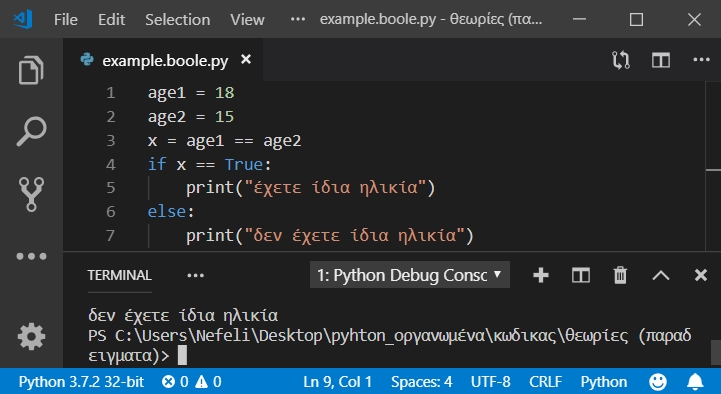
Παράδειγμα σύνταξης:

5 <= 7

True

4 >= 4.0

True



Άλγεβρα Boole

Η λογική Boole χρησιμοποιείται συνήθως μαζί με το if όταν πρέπει να εξεταστούν παραπάνω από μία συνθήκες.

Οι τρεις κύριες πράξεις της άλγεβρας Boole είναι η σύζευξη (**and**), η διάζευξη (**or**) και η άρνηση (**not**).

Η πράξη της σύζευξης (**and**) ελέγχει αν ισχύουν δύο διαφορετικές συνθήκες ταυτόχρονα. Αν ισχύουν και οι δυο συνθήκες τις αξιολογεί ως αληθείς (**True**). Σε κάθε άλλη περίπτωση τις αξιολογεί ως ψευδείς (**False**).

*Παράδειγμα σύνταξης:*

1 == 1 and 5 == 5

True

1 == 1 and 5 == 8

False

5 < 3 and 3 != 3

False

Η πράξη της διάζευξης (**or**) ελέγχει αν ισχύει τουλάχιστον μία από τις δύο συνθήκες. Αξιολογεί τις συνθήκες αυτές ως αληθείς (**True**) αν ισχύει έστω η μία από τις δύο (ή και οι δύο). Τις αξιολογεί ως ψευδείς (**False**), αν καμία από τις δύο τιμές δεν ισχύει.

*Παράδειγμα σύνταξης:*

1 == 1 or 5 == 5

True

1 == 1 or 5 == 8

True

5 < 3 or 3 != 3

False

Η πράξη της άρνησης (**not**) χρησιμοποιείται όταν θέλουμε να διατυπώσουμε μια συνθήκη αρνητικά. Πιο συγκεκριμένα: Το αποτέλεσμα όχι αληθές (**not True**) είναι ψευδές (**False**), ενώ το όχι ψευδές (**not False**) είναι αληθές (**True**).

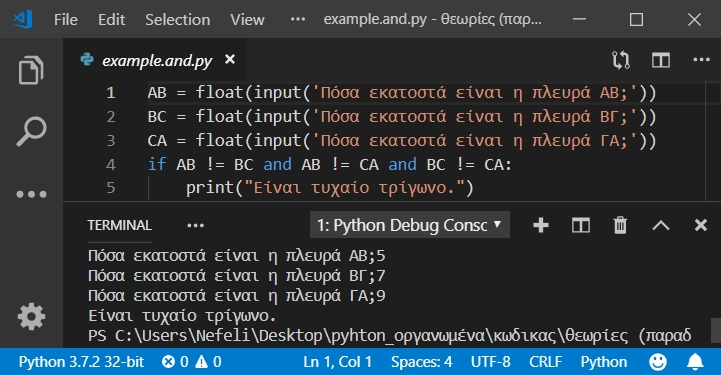
*Παράδειγμα σύνταξης:*

not 5 == 5

False

not 5 < 3

True



Βρόγχος while – while loops

Η δομή επανάληψης **while** χρησιμοποιείται όταν θέλουμε να γίνεται μια διαδικασία επαναλαμβανόμενα, για όσο ισχύει μια συνθήκη που έχουμε ορίσει.

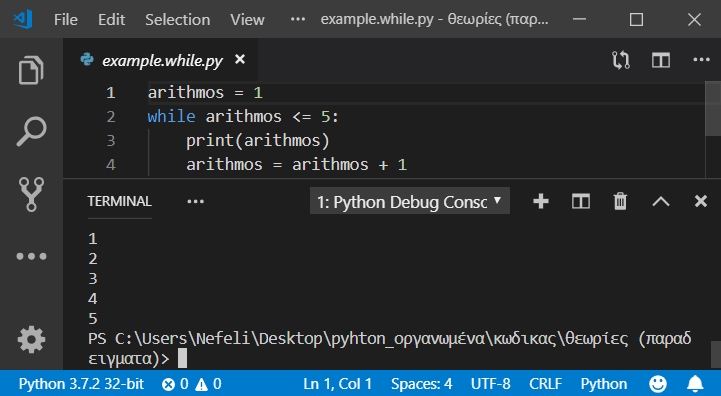
*Παράδειγμα σύνταξης:*

arithmos = 1

while arithmos <= 5:

print(arithmos)

arithmos = arithmos + 1



Λίστες - lists

Όπως μια συμβολοσειρά (**string**), η λίστα (**list**) είναι μία ακολουθία από τιμές. Σε μια συμβολοσειρά οι τιμές αυτές είναι χαρακτήρες-σύμβολα. Στις λίστες αυτές οι τιμές μπορούν να είναι πολλών τύπων (σύμβολα, αριθμοί, άλλες λίστες κ.α.). Οι τιμές που υπάρχουν μέσα σε μία λίστα ονομάζονται στοιχεία (elements) ή αντικείμενα (items).

*Παράδειγμα σύνταξης:*

lista = ["γάλα", "αυγά", "σοκολάτα", "τυρί", "δημητριακά"]

Αν και συνήθως μία λίστα περιέχει αντικείμενα με κοινά χαρακτηριστικά, είναι δυνατόν να περιέχει διαφορετικά είδη αντικειμένων (π.χ. strings, αριθμούς, λίστες).

*Παράδειγμα σύνταξης:*

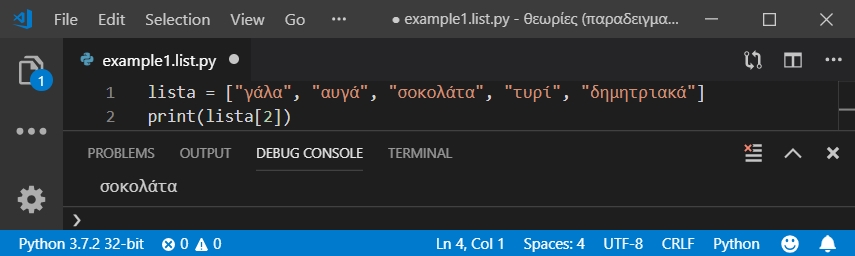
lista = ["γάλα", 8, "σοκολάτα", [0, 1, 2], 3.14]

Μπορούμε να επιλέξουμε να εμφανιστούν όλα τα αντικείμενα της λίστας, με την εντολή **print**:

print (lista)

Ή να επιλέξουμε να εμφανιστούν συγκεκριμένα αντικείμενα της λίστας:

print(lista[2])



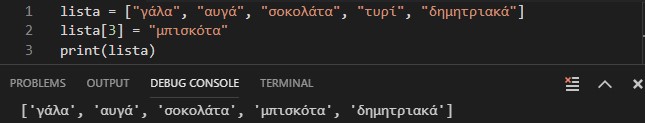
*ΣΗΜΕΙΩΣΗ:* Στον προγραμματισμό, η αρίθμηση ξεκινάει από το μηδέν και όχι από το ένα. Οπότε, στη συγκεκριμένη λίστα, το «γάλα» βρίσκεται στη θέση 0 της λίστας, τα «αυγά» στη θέση 1, η «σοκολάτα» στη θέση 2 κ.ο.κ..

Τα αντικείμενα μέσα σε μία λίστα μπορούν να αναπροσδιοριστούν.

*Παράδειγμα σύνταξης:*

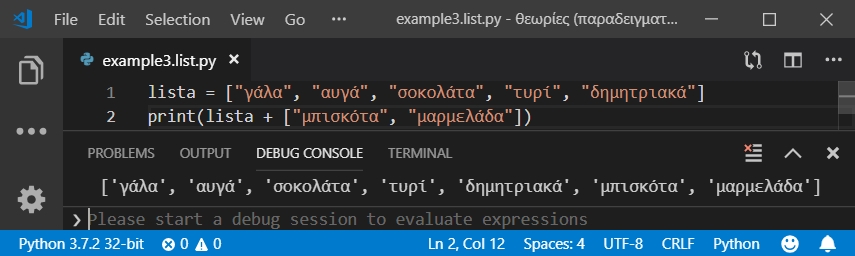
lista = ["γάλα", "αυγά", "σοκολάτα", "τυρί", "δημητριακά"]

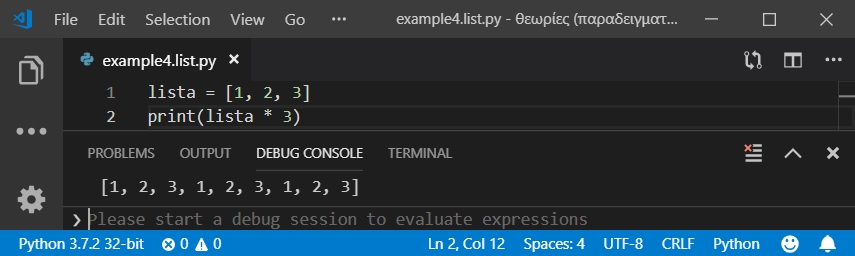
lista[3] = "μπισκότα"



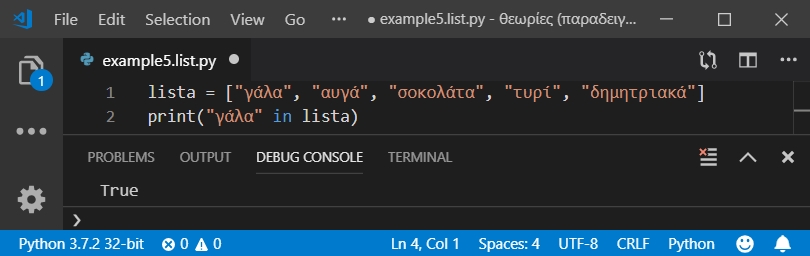
Τελεστές για τις λίστες – list operators

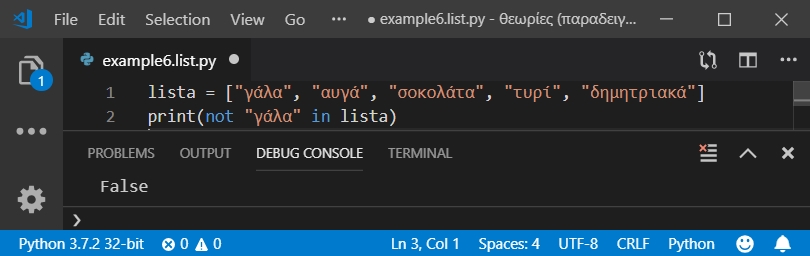
Οι λίστες μπορούν να προστεθούν και να πολλαπλασιαστούν:





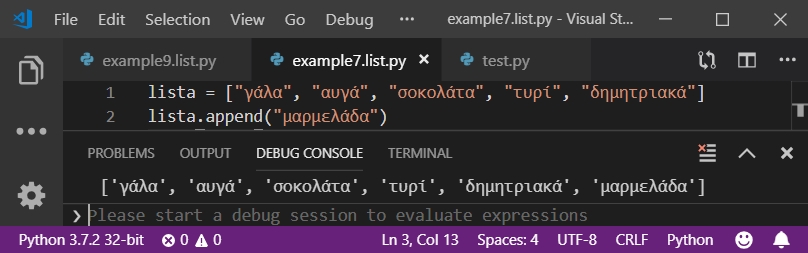
Μπορούμε να ελέγξουμε αν ένα στοιχείο υπάρχει στη λίστα, με τον τελεστή **in**, ή αν δεν υπάρχει, με τον τελεστή **not**. Ως αποτέλεσμα, o interpreter βγάζει True αν αυτό συμβαίνει ή False αν δεν συμβαίνει.



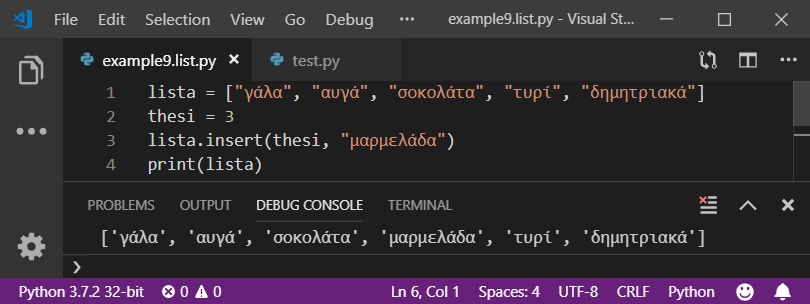


Λειτουργίες στις λίστες – list functions

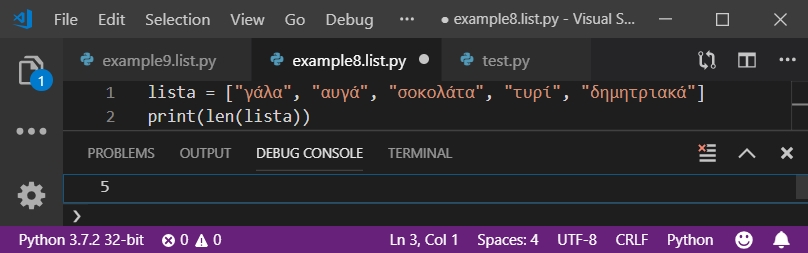
Με τη μέθοδο **append** προσθέτουμε στοιχεία στο τέλος της υπάρχουσας λίστας.



Η μέθοδος **insert** μας επιτρέπει, επίσης, να προσθέσουμε στοιχεία στην υπάρχουσα λίστα, σε συγκεκριμένη, όμως, θέση.



Για να ελέγξουμε τον αριθμό των στοιχείων που βρίσκονται στη λίστα χρησιμοποιούμε τη λειτουργία **len**.



Με τη μέθοδο **sum** βρίσκουμε το άθροισμα των στοιχείων που υπάρχουν σε μια λίστα.

lista = [1, 5, 8, 7, 9, 12.7]

x = sum(lista)

print(x)

η εντολή list()

Η λειτουργία **list** μετατρέπει μεταβλητές σε λίστες.

onoma = "Marina"

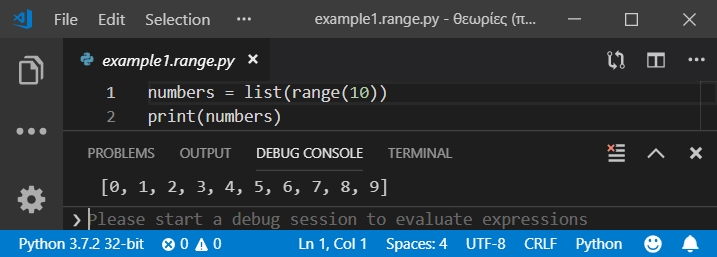
x = list(onoma)

print(x)

print(x[2])

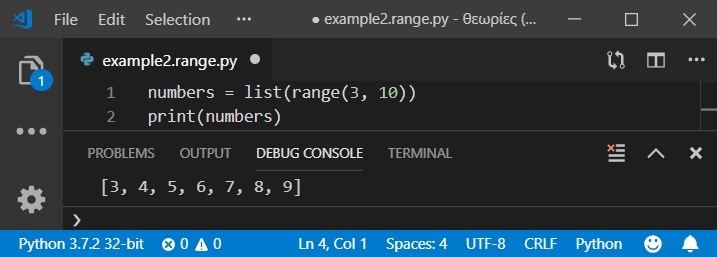
η εντολή range() - εύρος

Η λειτουργία **range** δημιουργεί ένα εύρος αριθμών, το οποίο για να το αξιοποιήσουμε πρέπει να το βάλουμε μέσα σε μία λίστα με τη εντολή list().

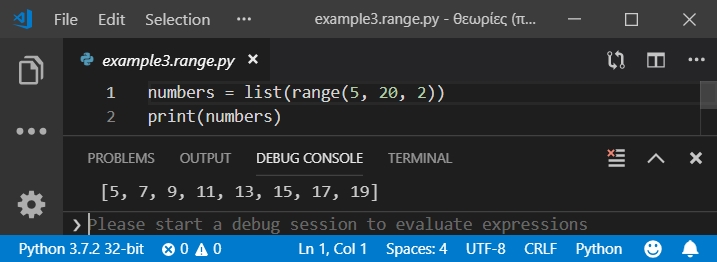


Στο παραπάνω παράδειγμα δημιουργήθηκε μία λίστα από το 0 μέχρι το 9.

Για να δημιουργήσουμε μία λίστα που να δεν ξεκινάει από το 0, αλλά από έναν άλλον αριθμό, χρησιμοποιούμε δύο περιορισμούς.



Μπορεί να χρησιμοποιηθεί ένας τρίτος περιορισμός, ο οποίος καθορίζει το διάστημα ανάμεσα στα στοιχεία του εύρους.



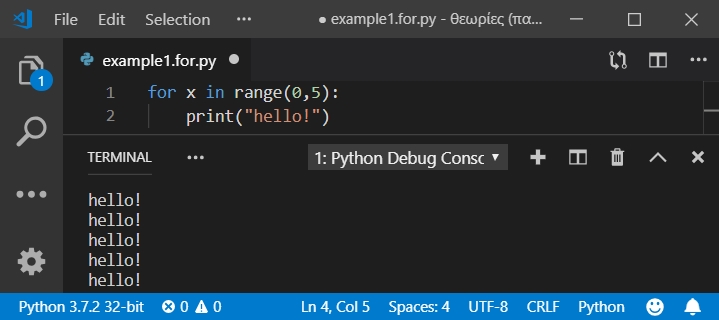
δομή επανάληψης for – for loops

Η δομή επανάληψης for είναι χρήσιμη για την επανάληψη μιας σειράς εντολών όταν είναι γνωστό εκ των προτέρων πόσες φορές θέλουμε να επαναληφθούν.

*Παράδειγμα σύνταξης:*

for x in range(0,5):

print("hello!")



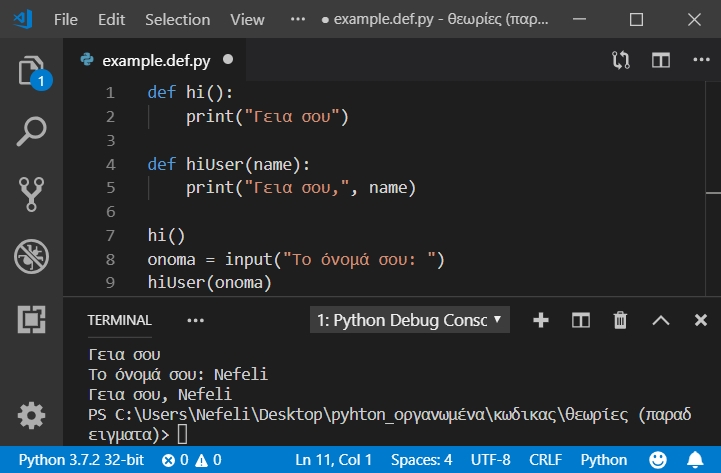
η εντολή def (define)

Η εντολή def μάς δίνει τη δυνατότητα να δημιουργήσουμε δικές μας εντολές, οι οποίες αποτελούν ουσιαστικά μια σειρά εντολών της Python. Με τον τρόπο αυτό μπορούμε να επαναχρησιμοποιούμε κώδικα απλώς καλώντας μια εντολή, αποφεύγοντας να τον ξαναγράφουμε εκ νέου κάθε φορά.

*Παράδειγμα σύνταξης:*

def hi():

print("Γεια σου")



Όταν δημιουργούμε μια λειτουργία με τη def πρέπει να ορίζουμε αν θα δέχεται παραμέτρους ή όχι.

Το σύστημα εισαγωγής (the import system)

Ένα από τα μεγαλύτερα πλεονεκτήματα της Python είναι η μεγάλη πρότυπη βιβλιοθήκη της, η οποία παρέχει «εργαλεία» κατάλληλα για διάφορες εργασίες. Πρόκειται ουσιαστικά για εισαγωγή (import) έτοιμου κώδικα που δίνει τη δυνατότητα να εκτελέσουμε εύκολα και γρήγορα ορισμένες ενέργειες.

Για παράδειγμα μπορούμε να βρούμε την τετραγωνική ρίζα ενός αριθμού χρησιμοποιώντας την εντολή sqrt, από τη βιβλιοθήκη math, χωρίς να χρειαστεί να γράψουμε εμείς τον αλγόριθμο εύρεσης της ρίζας.

Πρόσβαση σε μία βιβλιοθήκη έχουμε χρησιμοποιώντας την εντολή import.

Από τις βασικές βιβλιοθήκες της Python είναι η «math», η οποία περιέχει μαθηματικές λειτουργίες, και η «turtle», η οποία περιέχει γραφικά χελώνας (εμπνευσμένα από τη γλώσσα προγραμματισμού Logo).

*Παράδειγμα σύνταξης:*

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| import turtle  turtle.forward(200)  turtle.left(120)  turtle.forward(200)  turtle.left(120)  turtle.forward(200) | img_1318421504 |

Αν το όνομα της βιβλιοθήκης είναι σχετικά μεγάλο, συνηθίζεται να χρησιμοποιούμε την παρακάτω σύνταξη:

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| import turtle as t  t.forward(200)  t.left(120)  t.forward(200)  t.left(120)  t.forward(200) | img_1318421504 |

Ο παραπάνω κώδικας εισάγει, όπως φαίνεται, τη βιβλιοθήκη των γραφικών χελώνας και με τις κατάλληλες εντολές σχηματίζεται ένα τρίγωνο.

Είναι, όμως, δυνατόν να σχεδιάσουμε παραπάνω από ένα σχήματα, τα οποία πιθανόν να μην ενώνονται μεταξύ τους. Για να γίνει αυτό, μας βοηθούν οι εντολές penup() και pendown().

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| import turtle  x = 100  y = 50  turtle.forward(x)  turtle.penup()  turtle.forward(y)  turtle.pendown()  turtle.forward(x) | img_1354271504 |

# Γλωσσάρι

|  |  |  |
| --- | --- | --- |
| Αριθμητικοί Τελεστές:  +  -  \*  \*\*  /  //  % | Πρόσθεση  Αφαίρεση  Πολλαπλασιασμός  Ύψωση σε δύναμη  Διαίρεση  Πηλίκο ακέραιας διαίρεσης  Υπόλοιπο ακέραιας διαίρεσης | 5 + 3  5 – 3  5 \* 3  3 \*\* 2  4 / 2  5 // 2  5 % 2 |
| Τελεστές σύγκρισης:  ==  !=  > >=  <  <= | Ίσο  Διάφορο  Μεγαλύτερο Μεγαλύτερο ή ίσο  Μικρότερο  Μικρότερο ή ίσο | 2 == 2  2 != 3  3 > 2  x >= 3  2 < 3  x <= 3 |
| Λογικοί τελεστές:  and  or  not | Έλεγχος συνθηκών ως αληθείς ή ψευδείς  Σύζευξη  Διάζευξη  Άρνηση | 1 == 1 and 5 == 5  1 == 2 or 5 == 5  not 2 > 5 |
| Μεταβλητή (Variable)  string  integer τύποι μεταβλητών  float | Αποθήκευση δεδομένων στη μνήμη του υπολογιστή | x = 5  name = Nefeli  str(x) str(name)  int(x) int(5)  float(x) float(5) |
| Λίστα  in / not  append λειτουργίες σε  insert λίστες  len  sum | Αποθήκευση δεδομένων σε λίστα  Αναζήτηση στη λίστα  Εισαγωγή στο τέλος της λίστας  Εισαγωγή σε θέση στη λίστα  Αριθμός αντικειμένων της λίστας  Άθροισμα αντικειμένων της λίστας | lista = [1 , 5 , 9]  "9" in lista /  not "4" in lista  lista.append("5")  lista.insert(1,"3")  len(lista)  sum(lista) |
| input | Εισαγωγή δεδομένων από το χρήστη | input("Όνομα: ") |
| print () | Εξαγωγή δεδομένων στην οθόνη του υπολογιστή | print("hello") |
| if /  if…else | Έλεγχος συνθηκών και λήψη αποφάσεων | if x >= 18:  print("ενήλικος")  else:  print("ανήλικος") |
| while () | Επανάληψη μιας διαδικασίας για όσο ισχύει μια συνθήκη | while i <= 10:  print (x \* i)  i = i + 1 |
| for() | Υλοποίηση μιας σειράς εντολών για ορισμένο αριθμό επαναλήψεων | for x in range(0,3):  print(x) |
| list() | Μετατροπή μεταβλητής σε λίστα | name = nefeli  list(nefeli) |
| range() | Δημιουργία εύρους αριθμών | range(10)  range(5,10) |
| def() | Δημιουργία μιας εντολής που αποτελείται από μια σειρά εντολών | def triangle(x,y):  t.forward(x)  t.left(y)  t.forward(x)  t.left(y)  t.forward(x) |
| import | Εισαγωγή βιβλιοθήκης (έτοιμου κώδικα) | import.turtle |

*ΣΗΜΕΙΩΣΗ: Το παραπάνω γλωσσάρι αποτελεί ένα μικρό υποσύνολο του λεξιλογίου της Python 3.7 και περιέχει μόνο το λεξιλόγιο που χρησιμοποιείται στην παρούσα εργασία.*

# Δραστηριότητες

***Εισαγωγική δραστηριότητα***

*«Programming is usually taught by examples.»*

*-Niklaus Wirth*

Τα παιδιά εξοικειώνονται, αρχικά, με τη γλώσσα προγραμματισμού Python εισάγοντας (import) τη βιβλιοθήκη των γραφικών χελώνας (turtle). Σκοπός είναι να δημιουργήσουν έναν απλό αλγόριθμο για την σχεδίαση σχημάτων.

Εισαγωγική δραστηριότητα

Να σχεδιάσετε το βασικό σχέδιο δύο σπιτιών, το ένα δίπλα στο άλλο.

Τα σπίτια να αποτελούνται από ένα τετράγωνο (διαστάσεων 100x100 pixels) και ένα ισόπλευρο τρίγωνο (με μήκος πλευρών 100 pixels και άνοιγμα γωνιών 60 μοίρες) το καθένα.

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| import turtle as t  def spiti():  t.forward(200)  t.left(90)  t.forward(200)  t.left(90)  t.forward(200)  t.left(90)  t.forward(200)  def skepi():  t.right(30)  t.forward(200)  t.right(120)  t.forward(200)  spiti()  t.penup()  t.left(180)  t.forward(200)  t.pendown()  skepi()  t.penup()  t.right(30)  t.forward(200)  t.left(90)  t.forward(20)  t.pendown()  spiti()  t.penup()  t.left(180)  t.forward(200)  t.pendown()  skepi() | σπιτια |

***Επίλυση απλών μαθηματικών προβλημάτων***

Στις παρακάτω δραστηριότητες οι μαθητές θα κληθούν να επιλύσουν απλά μαθηματικά προβλήματα, χρησιμοποιώντας βασικά στοιχεία της γλώσσας, όπως αναφέρονται παραπάνω. Στόχος είναι να εκφράσουν σε κώδικα τον κατάλληλο κάθε φορά αλγόριθμο προκειμένου να επιλύσουν το πρόβλημα.

Δραστηριότητα 1

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει τον Μέσο Όρο τριών αριθμών, που θα εισάγει ο χρήστης.

*Τρόπος σκέψης:*

*Α) Ορίζουμε τις μεταβλητές. Δημιουργούμε, δηλαδή, τρεις (3) μεταβλητές (a, b, c), η καθεμία από τις οποίες έχει τιμή τα δεδομένα που εισάγει ο χρήστης, όπως ορίζεται με την εντολή input. Είναι σημαντικό να ορίσουμε τον τύπο της μεταβλητής, ώστε ο Η/Υ να αναγνωρίζει τι είδους δεδομένα εισάγονται. Χρησιμοποιούμε τον τύπο* float *για να καταστήσουμε σαφές ότι τα δεδομένα που εισάγονται είναι αριθμοί και για να δώσουμε την ευελιξία στο χρήστη να εισάγει τόσο ακέραιους όσο και δεκαδικούς αριθμούς.*

Β) *Με την εντολή* print *εμφανίζουμε στην οθόνη του Η/Υ τον Μ.Ο. των αριθμών που εισήχθησαν χρησιμοποιώντας την κατάλληλη μαθηματική εξίσωση* ( (a + b + c)/3) ). *Βάζουμε κόμμα ( , ) ανάμεσα στη φράση (string) και στη μαθηματική πράξη που περιλαμβάνονται μέσα στο print για να τα διαχωρίσουμε, επειδή πρόκειται για διαφορετικούς τύπους δεδομένων (*string *και* float*)*

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής*

*Ενδεικτικός τρόπος επίλυσης:*

print("Βρείτε το μέσο όρο 3 αριθμών")

a = float(input("Εισάγετε τον πρώτο αριθμό:"))

b = float(input("Εισάγετε τον δεύτερο αριθμό:"))

c = float(input("Εισάγετε τον δεύτερο αριθμό:"))

print("Ο μέσος όρος των τριών αριθμών είναι:" , (a + b + c)/3)

Πώς θα βρίσκατε τον Μέσο Όρο αριθμών αν έπρεπε να χρησιμοποιήσετε λίστα;

list = []

for i in range(0,3):

x = float(input("Εισάγετε έναν αριθμό:"))

list.append(x)

print(sum(list)/len(list))

Δραστηριότητα 2

Να φτιάξετε ένα πρόγραμμα που θα ελέγχει αν ο αριθμός που εισάγει ο χρήστης είναι μονός ή ζυγός.

*Τρόπος σκέψης:*

*Α) Ορίζουμε μια μεταβλητή (x),* για να αποθηκεύσουμε τα δεδομένα που χρειάζονται. Ορίζουμε ότι ο τύπος της μεταβλητής είναι ακέραιος αριθμός (int)*. Καθιστούμε σαφές, με την εντολή* input*, ότι τα δεδομένα που αποθηκεύει η μεταβλητή θα το εισάγει ο χρήστης.*

*Σημείωση: Ορίζουμε μεταβλητή τύπου* int *καθώς μόνο οι ακέραιοι αριθμοί μπορούν να χαρακτηριστούν ως μονοί ή ζυγοί*

*Β) Έχοντας στο νου τα χαρακτηριστικά των μονών και των ζυγών αριθμών, ορίζουμε τη συνθήκη σύμφωνα με την οποία ένας αριθμός είναι μονός ή ζυγός. Αν ο αριθμός (x) που εισάγει ο χρήστης διαιρεθεί με το δύο (2) και έχει υπόλοιπο μηδέν (0) (*x % 2 == 0*) εμφανίζουμε (*print*) στην οθόνη του Η/Υ το κείμενο «Ο αριθμός είναι ζυγός». Σε διαφορετική περίπτωση εμφανίζουμε (*print*) στην οθόνη του Η/Υ το κείμενο «Ο αριθμός είναι μονός».*

*Θα χρειαστείτε: input, print, if, πράξεις, ορισμό τύπου μεταβλητής*

*Ενδεικτικός τρόπος επίλυσης:*

x = int(input("Εισάγετε έναν αριθμό:"))

if x % 2 == 0:

print("Ο αριθμός είναι ζυγός.")

else:

print("ο αριθμός είναι μονός.")

Δραστηριότητα 3

Να φτιάξετε ένα πρόγραμμα που να μετατρέπει σε δέκατα, εκατοστά, χιλιοστά και χιλιόμετρα την απόσταση που εισάγει ο χρήστης σε μέτρα.

*Τρόπος σκέψης:*

*Α) Ορίζουμε μια μεταβλητή (x) τύπου* float *της οποία την τιμή εισάγει ο χρήστης (*input*).*

*Σημείωση: Ο τύπος της μεταβλητής ορίζεται ως* float*, έτσι ώστε ο χρήστης να έχει τη δυνατότητα να εισάγει τόσο ακέραιους, με τη μορφή δεκαδικού, αριθμούς (π.χ. 35.0) όσο και δεκαδικούς αριθμούς (π.χ. 35.5), καθώς η απόσταση δεν είναι απαραίτητο να είναι ακέραιος αριθμός.*

*Β) Ορίζουμε μια δεύτερη μεταβλητή (y), της οποίας η τιμή είναι το αποτέλεσμα μιας πράξης, και στη συνέχεια εμφανίζεται (*print*) το αποτέλεσμά της. Η μεταβλητή αυτή αποθηκεύει, αρχικά το αποτέλεσμα της πράξης* x \* 10 *και κάθε φορά επαναπροσδιορίζεται για να αποθηκεύσει τις πράξεις* x \* 100 , x \* 1000 , x / 1000 *και έτσι να μετατρέψει την απόσταση σε μέτρα που εισάγει ο χρήστης σε δέκατα, εκατοστά, χιλιοστά και χιλιόμετρα αντίστοιχα. Το αποτέλεσμα που εμφανίζεται κάθε φορά ορίζεται ως «*string*», ώστε να μπορεί να εμφανιστεί μία ολοκληρωμένη πρόταση που περιέχει και αριθμό και λέξεις* (str(y) + " δέκατα")*. Οι αριθμοί αναγνωρίζονται πλέον ως λέξη, καθώς δε θα μπορούσαν να «προστεθούν» δύο διαφορετικού τύπου μεταβλητές.*

*Θα χρειαστείτε: input, print, πράξεις,* *ορισμό τύπου μεταβλητής*

*Ενδεικτικός τρόπος επίλυσης:*

x = float(input ("Εισάγετε σε μέτρα την απόσταση που θέλετε να μετατρέψετε:"))

y = x \* 10

print (str(y) + " δέκατα")

y = x \* 100

print (str(y) + " εκατοστά")

y = x \* 1000

print (str(y) + " χιλιοστά")

y = x / 1000

print (str(y) + " χιλιόμετρα")

Δραστηριότητα 4

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει την τιμή ενός προϊόντος μετά από έκπτωση 30% στην αρχική τιμή.

*Τρόπος σκέψης:*

*Α) Ορίζουμε μια μεταβλητή (x) τύπου* float*, της οποίας την τιμή εισάγει ο χρήστης (*input*).*

*Σημείωση: Χρησιμοποιείται μεταβλητή τύπου* float*, καθώς η τιμή ενός προϊόντος δεν είναι πάντα ακέραιος αριθμός, οπότε πρέπει να μπορεί να εισαχθεί και ακέραιος αριθμός (με τη μορφή δεκαδικού) και δεκαδικός αριθμός.*

*Β) Ορίζουμε μια νέα μεταβλητή (y), της οποίας η τιμή είναι το αποτέλεσμα της μαθηματικής πράξης που υπολογίζει την τιμή του προϊόντος που εισάγει ο χρήστης μετά την έκπτωση (*((100 - 30) \* x) / 100*).*

*Γ) Εμφανίζουμε (*print*) στην οθόνη του Η/Υ την τιμή της μεταβλητής y.*

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής*

*Ενδεικτικός τρόπος επίλυσης:*

x = float(input ("Ποια είναι η αρχική τιμή του προϊόντος;"))

y = ((100 - 30) \* x) / 100

print (y)

Δραστηριότητα 5

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει το εμβαδόν ενός ορθογωνίου τριγώνου, εισάγοντας το μήκος των δύο κάθετων πλευρών του.

*Τρόπος σκέψης:*

*Α) Έχοντας στο νου τον μαθηματικό τύπο για την εύρεση εμβαδού ορθογωνίου τριγώνου, ορίζουμε δύο μεταβλητές (a, b) τύπου* float*, των οποίων τις τιμές (μήκος πρώτης κάθετης και δεύτερης κάθετης πλευράς αντίστοιχα) εισάγει ο χρήστης.*

*Σημειώσεις: Χρησιμοποιείται μεταβλητή τύπου* float*, καθώς το μήκος μιας πλευράς μπορεί να είναι ακέραιος ή δεκαδικός αριθμός.*

*Β) Ορίζουμε μια τρίτη μεταβλητή (x), της οποίας η τιμή είναι το αποτέλεσμα της μαθηματικής πράξης που υπολογίζει το εμβαδόν ενός ορθογωνίου τριγώνου (*(a \* b) / 2*).*

*Γ) Εμφανίζουμε (*print*) την τιμή της μεταβλητής x.*

*Σημείωση: Το αποτέλεσμα που εμφανίζεται στην οθόνη αποτελείται από δύο φράσεις που ενώνονται με το σύμβολο «+». Η μία ορίζεται μέσα στα εισαγωγικά και η άλλη ορίζεται με τη μεταβλητή τύπου* string *(το εμβαδόν, που εδώ αναγνωρίζεται ως λέξη, για να μπορεί να συμπεριληφθεί στην πρόταση, διαφορετικά θα έπρεπε να ενώσουμε τις δυο μεταβλητές μέσα στο* print *με το κόμμα (,)).*

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής*

*Ενδεικτικός τρόπος επίλυσης:*

a = float(input ("Εισάγετε το μήκος της πρώτης κάθετης πλευράς:"))

b = float(input ("Εισάγετε το μήκος της δεύτερης κάθετης πλευράς:"))

x = (a \* b) / 2

print ("Το εμβαδόν του τριγώνου είναι: " + str(x))

Δραστηριότητα 6

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει τη διάμετρο και το μήκος ενός κύκλου, όταν ο χρήστης εισάγει την ακτίνα του.

*Τρόπος σκέψης:*

*Α) Έχοντας στο νου τους τύπους για την εύρεση της διαμέτρου και του μήκους του κύκλου, ορίζουμε μια μεταβλητή (x) τύπου* float*, της οποίας την τιμή (το μήκος της ακτίνας) εισάγει ο χρήστης.*

*Σημείωση: Χρησιμοποιείται η μεταβλητή* float *γιατί το μήκος της ακτίνας δύναται να είναι δεκαδικός αριθμός.*

*Β) Εμφανίζουμε (*print*) το επιθυμητό αποτέλεσμα σε δύο προτάσεις. Στην πρώτη πρόταση εμφανίζεται η διάμετρος του κύκλου* ("Η διάμετρος του κύκλου είναι:" , (x \* 2)) *και στην άλλη το μήκος του κύκλου* ("Το μήκος του κύκλου είναι:" , (2 \* x \* pi))*.*

*Σημείωση: Οι φράσεις και το αποτελέσματα των πράξεων που περιέχονται στην πρόταση διαχωρίζονται μεταξύ τους με κόμμα (,), γιατί είναι διαφορετικοί τύποι μεταβλητών.*

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής*

*Ενδεικτικός τρόπος επίλυσης:*

x = float(input("Εισάγετε την ακτίνα του κύκλου: "))

pi = 3.14159

print ("Η διάμετρος του κύκλου είναι:" , (x \* 2))

print ("Το μήκος του κύκλου είναι:" , (2 \* x \* pi))

Δραστηριότητα 7

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει την προπαίδεια ενός ακέραιου αριθμού.

*Τρόπος σκέψης:*

*Α) Ορίζουμε μια μεταβλητή (x) τύπου* int*, της οποίας την τιμή εισάγει ο χρήστης.  
Σημείωση: Χρησιμοποιείται μεταβλητή τύπου* int*, γιατί συνήθως ζητείται η προπαίδεια ακέραιων αριθμών.*

*Β) Ορίζουμε τον τρόπο που θα πολλαπλασιάζεται ο αριθμός που εισάγει ο χρήστης ώστε να εμφανιστεί η προπαίδειά του:*

*Η μέτρηση ξεκινάει από τον αριθμό μηδέν (* i=0 *). Ο Η/Υ εμφανίζει τον αριθμό που του δίνεται και τον πολλαπλασιάζει αρχικά με το 0 (* print (x \* i) *). Στη συνέχεια προστίθεται 1 μονάδα στον πολλαπλασιαστή (* i = i + 1 *) και ο πολλαπλασιαστέος πολλαπλασιάζεται τώρα με τον καινούριο αριθμό (* print (x \* i) *). Η διαδικασία επαναλαμβάνεται μέχρι ο πολλαπλασιαστής να γίνει 10, δηλαδή όσο ισχύει ότι ο πολλαπλασιαστής είναι μικρότερος ή ίσος του 10 (* while i <= 10 *).*

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής, συγκρίσεις, while ή for/in*

*Ενδεικτικός τρόπος επίλυσης:*

x = int(input("Υπολόγισε την προπαίδεια του αριθμού: "))

i=0

while i <= 10:

print (x \* i)

i = i + 1

Δραστηριότητα 8

Να φτιάξετε ένα πρόγραμμα όπου ο χρήστης θα εισάγει μια ηλικία και θα ελέγχεται αν είναι ενήλικος. Σε περίπτωση που είναι ανήλικος, να υπολογίζεται σε πόσα χρόνια θα ενηλικιωθεί.

*Τρόπος σκέψης:*

*Α) Ορίζουμε μία μεταβλητή (x) τύπου* int*, της οποίας την τιμή (ηλικία) εισάγει ο χρήστης.*

*Σημείωση: Χρησιμοποιείται μεταβλητή τύπου* int *καθώς ζητείται από το χρήστη να εισάγει την ηλικία του σε χρόνια (ακέραιος αριθμός) και όχι σε χρόνια και μήνες (δεκαδικός αριθμός).*

*Β) Ορίζονται οι συνθήκες που ισχύουν όταν είναι κανείς ενήλικος ή ανήλικος:*

*Αφού ο χρήστης εισάγει την ηλικία του, ο Η/Υ ελέγχει αν ο αριθμός είναι μεγαλύτερος ή ίσος του 18 (* if x >= 18 *), δηλαδή αν ο χρήστης είναι ενήλικος και εμφανίζει την ανάλογη φράση (* print ("Είστε ενήλικος.") *).*

*Σε άλλη περίπτωση (* else *), δηλαδή αν δεν ισχύει η παραπάνω υπόθεση, δημιουργείται μία μεταβλητή (y) τύπου* int*, της οποίας η τιμή είναι το αποτέλεσμα της αφαίρεση της ηλικίας του χρήστη από το 18 (* y = int(18 - x) *), που θωρείται πως είναι η ενηλικίωση, για να υπολογίσει σε πόσα χρόνια ο χρήστης θα είναι ενήλικος. Στη συνέχεια εμφανίζεται το αποτέλεσμα (* print ("Χρειάζεστε ακόμη " + str(y) + " χρόνια για να ενηλικιωθείτε.") *).*

*Σημείωση: Το αποτέλεσμα που εμφανίζεται στην οθόνη αποτελείται από δύο φράσεις που ενώνονται με το σύμβολο «+». Η μία ορίζεται μέσα στα εισαγωγικά και η άλλη ορίζεται με τη μεταβλητή τύπου* string*.*

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής, if, else, συγκρίσεις*

*Ενδεικτικός τρόπος επίλυσης:*

x = int(input ("Εισάγετε ηλικία:"))

if x >= 18:

print ("Είστε ενήλικος.")

else:

y = int(18 - x)

print ("Χρειάζεστε ακόμη " + str(y) + " χρόνια για να ενηλικιωθείτε.")

Δραστηριότητα 9

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει το ελάχιστο κοινό πολλαπλάσιο δύο αριθμών.

*Τρόπος σκέψης:*

*Α) Ορίζουμε δύο μεταβλητές ( x , y ) τύπου* int*, των οποίων τις τιμές εισάγει ο χρήστης και μία μεταβλητή ( i ), όπου ορίζουμε από που θα ξεκινήσει η αρίθμηση.*

*Β) Ορίζουμε τις διαδικασίες που θα γίνουν σε τρεις διαφορετικές περιπτώσεις: Αν ο πρώτος αριθμός ( x ) είναι μεγαλύτερος από τον δεύτερο αριθμό ( y ), αν ο δεύτερος αριθμός ( y ) είναι μεγαλύτερος από τον πρώτο αριθμό ( x ), αν οι δύο αριθμοί είναι ίσοι.*

*Αν x > y, το x πολλαπλασιάζεται κάθε φορά με το i μέχρι να το αποτέλεσμα του πολλαπλασιασμού να μην είναι διάφορο του μηδενός (* while (x \* i) % y != 0: i = i + 1 *) και στη συνέχεια εμφανίζεται το αποτέλεσμα στην οθόνη (* print("Το ΕΚΠ είναι το: " + str(x \* i)) *) . Αντίστοιχη είναι η διαδικασία, αν y > x.*

*Αν x == y, το ΕΚΠ είναι ο ίδιος αριθμός και εμφανίζεται στην οθόνη οποιοσδήποτε από τους δυο αριθμούς (* print("To ΕΚΠ είναι το: " + str(x)) ή print("To ΕΚΠ είναι το: " + str(y)) *).*

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής, if, συγκρίσεις*

*Ενδεικτικός τρόπος επίλυσης:*

x = int(input("Eισάγετε τον πρώτο αριθμό: "))

y = int(input("Eισάγετε τον δεύτερο αριθμό: "))

i = 1

if x > y:

while (x \* i) % y != 0:

i = i + 1

print("Το ΕΚΠ είναι το: " + str(x \* i))

if y > x:

while (y \* i) % x != 0:

i = i + 1

print("Το ΕΚΠ είναι το: " + str(y \* i))

if x == y:

print("To ΕΚΠ είναι το: " + str(x))

Δραστηριότητα 10

Να φτιάξετε ένα πρόγραμμα που να κατατάσσει σε φθίνουσα σειρά τρεις αριθμούς που θα εισάγει ο χρήστης.

*Τρόπος σκέψης:*

*Α) Ορίζουμε τρεις μεταβλητές ( x , y , z ) τύπου* int*,των οποίων τις τιμές εισάγει ο χρήστης*

*Β) Ορίζουμε τρεις διαφορετικές υποθέσεις (if, elif, else), οι οποίες ενέχουν άλλες συνθήκες if (εμφωλευμένο if), ώστε να γίνει ο κατάλληλος έλεγχος και να εμφανιστούν οι αριθμοί σε σωστή σειρά.*

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής, if/elif/else, συγκρίσεις*

*Ενδεικτικός τρόπος επίλυσης:*

x = int(input("Εισάγετε τον πρώτο αριθμό: "))

y = int(input("Εισάγετε τον δεύτερο αριθμό: "))

z = int(input("Εισάγετε τον τρίτο αριθμό: "))

if x > y and x > z:

if y > z:

print (str (x) + ">" + str(y) + ">" + str(z))

if y < z:

print (str (x) + ">" + str(z) + ">" + str(y))

elif y > x and y > z:

if x > z:

print (str (y) + ">" + str(x) + ">" + str(z))

if x < z:

print (str (y) + ">" + str(z) + ">" + str(x))

else:

if x > y:

print (str (z) + ">" + str(x) + ">" + str(y))

if x < y:

print (str (z) + ">" + str(y) + ">" + str(x))

Δραστηριότητα 11

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει πόσο χρόνο (σε ώρες και λεπτά) θα χρειαστεί ένα όχημα για να διανύσει μια συγκεκριμένη απόσταση (σε χλμ), γνωρίζοντας με πόσα χλμ/ώρα κινείται (κατά μέσο όρο).

*Τρόπος σκέψης:*

*Α) Ορίζουμε δύο μεταβλητές (apostasi, taxitita) τύπου* float*, των οποίων τις τιμές εισάγει ο χρήστης, και δυο μεταβλητές (xOres , xlepta), των οποίων η τιμή είναι το αποτέλεσμα της διαίρεσης apostasi / taxitita (δηλαδή το ζητούμενο του προβλήματος) και το αποτέλεσμα της ίδιας διαίρεσης σε λεπτά (xOres \* 60) αντίστοιχα.*

*Για να εμφανιστεί το ζητούμενο του προβλήματος, ο χρόνος που θα χρειαστεί ένα όχημα για να διανύσει μια συγκεκριμένη απόσταση, σε μια ολοκληρωμένη πρόταση:*

*Β) Δημιουργούμε μια συνθήκη, όπου αν ο χρόνος σε λεπτά είναι περισσότερος ή ίσος με 60’, δηλαδή περισσότερο από 1 ώρα (* if xlepta>=60 *), χρησιμοποιούνται οι πράξεις* xlepta/60 *και* xlepta%60*, για να διαχωριστούν οι ώρες από τα λεπτά και να συμπεριληφθούν μαζί σε μία πρόταση (* print(int(xlepta/60), "ώρες και", int(xlepta%60), "λεπτά.") *).*

*Σε άλλη περίπτωση (*else*), δηλαδή αν πρόκειται για χρόνο λιγότερο της μιας ώρας, εμφανίζεται απλά η τιμή της μεταβλητής xlepta (* print(int(xlepta), "λεπτά" *).*

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής, if/elif/else, συγκρίσεις*

*Ενδεικτικός τρόπος επίλυσης:*

apostasi = float(input("Πόση απόσταση θα διανύσει το όχημα (σε χλμ);"))

taxitita = float(input("Ποιά είναι η ταχύτητα του οχήματος (σε χλμ/ώρα);"))

xOres = apostasi/taxitita

xlepta = xOres \* 60

if xlepta>=60:

print(int(xlepta/60), "ώρες και", int(xlepta%60), "λεπτά.")

else:

print(int(xlepta), "λεπτά")

Δραστηριότητα 12

Να φτιάξετε ένα πρόγραμμα που να δίνει σε κάθε φοιτητή έναν αριθμό μητρώου. Το πρόγραμμα θα ζητάει από τον φοιτητή να καταχωρήσει τα στοιχεία του (όνομα, επώνυμο, έτος εισαγωγής στο πανεπιστήμιο) και θα του δίνει έναν μοναδικό αριθμό μητρώου. Ο αριθμός μητρώου θα αποτελείται από το έτος εισαγωγής στο πανεπιστήμιο και από τον αύξοντα αριθμο (σειρά με την οποία γραφτηκε) στο τέλος του έτους εισαγωγής.

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής, if/elif/else, συγκρίσεις*

*Τρόπος σκέψης:*

*Σε αυτή τη δραστηριότητα, επειδή δεν ξέρουμε εκ των προτέρων πόσοι φοιτητές θα καταχωρηθούν, δηλαδή πόσες φορές θα τρέξει ο κώδικας, θα χρειαστεί να δημιουργήσουμε έναν βρόγχο επανάληψης (while loop).*

*Α) Δημιουργούμε μια μεταβλητή τύπου Boolean και την καταχωρούμε αρχικά αληθή (*neaKatahorisi = True*)*

*B) Ορίζουμε μια μεταβλητή (* i = 1 *), όπου αποθηκεύεται ο αύξοντας αριθμός (η σειρά με την οποία γράφτηκε ο φοιτητής).*

*Γ) Όσο η μεταβλητή neaKatahorisi είναι αληθής (* while neaKatahorisi==True *) θα επαναλαμβάνεται ο κώδικας που καταχωρεί τους φοιτητές.*

*Δημιουργούμε 2 μεταβλητές (onoma, eponimo) για να αποθηκεύουμε τα αντίστοιχα στοιχεία του φοιτητή και μια μεταβλητή (etos) τύπου int για να αποθηκεύουμε το έτος εγγραφής. Τις τιμές των μεταβλητών τις εισάγει ο χρήστης.*

*Ορίζεται μια μεταβλητή (AM) η οποία έχει ως τιμή το έτος εγγραφής, αφού μετατραπεί σε str, και τον αύξοντα αριθμό, αφού μετατραπεί σε str, έτσι ώστε να μην προσθέτονται οι αριθμοί αλλά να γράφονται απλώς δίπλα δίπλα ως κείμενο (* AM = str(etos) + str(i) *).*

*Εμφανίζονται τα στοιχεία του φοιτητή (* print(onoma , eponimo, AM) *) και ο αύξοντας αριθμός αυξάνεται κατά 1 (* i = i + 1 *), ώστε να έχει διαφορετική τιμή αν χρειαστεί να καταχωρήσουμε νέο φοιτητή.*

*Δημιουργούμε μια μεταβλητή με την οποία ζητάμε από το χρήστη να καταχωρήσει αν επιθυμεί να κάνει άλλη καταχώρηση (* telosKatahorisis = input("Εισαγωγή και άλλου φοιτητή; (ΝΑΙ/ΟΧΙ)") *). Αν η τιμή που θα εισάγει ο χρήστης είναι «ΟΧΙ» (* if telosKatahorisis == "ΟΧΙ" *), η τιμή της μεταβλητής neaKatahorisi γίνεται ψευδής (* neaKatahorisi = False *), οπότε ο κώδικας που βρίσκεται μέσα στο βρόγχο επανάληψης (while) δεν εκτελείται πλέον και το πρόγραμμα τερματίζει.*

*Ενδεικτικός τρόπος επίλυσης:*

neaKatahorisi = True

i = 1

while neaKatahorisi==True:

onoma = input("Όνομα φοιτητή:")

eponimo = input("Επώνυμο φοιτητή:")

etos = int(input("Έτος πρώτης εγγραφής:"))

AM = str(etos) + str(i)

print(onoma , eponimo, AM)

i = i + 1

telosKatahorisis = input("Εισαγωγή και άλλου φοιτητή; (ΝΑΙ/ΟΧΙ)")

if telosKatahorisis == "ΟΧΙ":

neaKatahorisi = False

***Επίλυση σύνθετων μαθηματικών προβλημάτων***

Αφού οι μαθητές ολοκλήρωσαν τις παραπάνω δραστηριότητες με επιτυχία, θεωρούμε ότι έχουν αποκτήσει μια εξοικέιωση με το πολύ βασικό λεξιλόγιο και τη σύνταξη της Python και είναι σε θέση να εκφράσουν στη γλώσσα αυτή έναν απλό αλγόριθμο.

Στις δραστηριότητες που ακολουθούν, θα χρησιμοποιήσουμε κάποιες επιπλέον εντολές της Python και θα επικεντρωθούμε στην έκφραση πιο σύνθετων μαθηματικών αλγορίθμων.

Δραστηριότητα 13

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει το κόστος αγοράς ενός διαμερίσματος (με το ΦΠΑ 24% επί της τελικής τιμής) ανάλογα με την περιοχή που βρίσκεται και την έκτασή του σε τ.μ. (Κέντρο 700€/τ.μ., Αγία Σοφία 600€/τ.μ, Ρίο 800€/τ.μ).

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής, if, συγκρίσεις, άλγεβρα Boole*

*Τρόπος σκέψης:*

*Α) Ορίζουμε 2 μεταβλητές (ektasi , perioxi) τύπου* float *(η μεταβλητή ektasi αποθηκευει τ.μ., δηλαδή αριθμό) και* string *(η μεταβλητή perioxi αποθηκευει το όνομα της περιοχής που βρίσκεται το ακίνητο, δηλαδή λέξη), των οποίων τις τιμές εισάγει ο χρήστης.*

*Β) Χρησιμοποιούμε 3 ενδεικτικές περιοχές της Πάτρας (κέντρο, Αγία Σοφία, Ρίο) και υπολογίζουμε το κόστος των ακινήτων σε κάθε περιοχή.*

*Υποθέτουμε 3 πιθανές διατυπώσεις της περιοχής από το χρήστη (* if perioxi == "ΚΕΝΤΡΟ" or perioxi =="Κέντρο" or perioxi =="κέντρο": *).*

*Ορίζουμε μια μεταβλητή (* proForou *), τιμή της οποίας είναι η τιμή του ακινήτου ανά τ.μ. ανάλογα με την περιοχή (* proForou = ektasi \* 700 *).*

*Ορίζουμε μια μεταβλητή (* foros *), τιμή της οποίας είναι ο φόρος 24% (* foros = (proForou \* 24) / 100 *).*

*Ορίζουμε μια μεταβλητή ( meta*Forou *), τιμή της οποίας είναι το άθροισμα των 2 παραπάνω μεταβλητών (* metaForou = proForou + foros *), δηλαδή η συνολική τιμή του ακινήτου.*

*Εμφανίζεται η τιμή του ακινήτου (* print ("Η τιμή του ακινήτου με τον φόρο είναι €" + str(metaForou)) *).*

*Η ίδια διαδικασία γίνεται και για τις άλλες δύο περιοχές (* elif perioxi == "ΑΓΙΑ ΣΟΦΙΑ" or perioxi =="Αγία Σοφία" or perioxi =="αγία σοφία": / elif perioxi == "ΡΙΟ" or perioxi =="Ρίο" or perioxi =="ρίο": *).*

*Σε άλλη περίπτωση (*else*), αν ο χρήστης εισάγει κάποια περιοχή που δεν συμπεριλαμβάνεται στο πρόγραμμα, εμφανίζεται το ανάλογο μήνυμα (* print("Η περιοχή δε βρέθηκε! \n Δοκιμάστε: ΚΕΝΤΡΟ, ΑΓΙΑ ΣΟΦΙΑ, ΡΙΟ")  *).*

*Σημείωση: Τα σύμβολα «*\n*» χρησιμοποιούνται ώστε το κείμενο που βρίσκεται μετά από αυτά να εμφανίζεται στην επόμενη γραμμή.*

*Ενδεικτικός τρόπος επίλυσης:*

ektasi = float(input("Πόσα τ.μ. είναι το ακίνητο;"))

perioxi = str(input("Σε ποια περιοχή βρίσκεται το ακίνητο;"))

if perioxi == "ΚΕΝΤΡΟ" or perioxi =="Κέντρο" or perioxi =="κέντρο":

proForou = ektasi \* 700

foros = (proForou \* 24) / 100

metaForou = proForou + foros

print ("Η τιμή του ακινήτου με τον φόρο είναι €" + str(metaForou))

elif perioxi == "ΑΓΙΑ ΣΟΦΙΑ" or perioxi =="Αγία Σοφία" or perioxi =="αγία σοφία":

proForou = ektasi \* 600

foros = (proForou \* 24) / 100

metaForou = proForou + foros

print ("Η τιμή του ακινήτου με τον φόρο είναι €" + str(metaForou))

elif perioxi == "ΡΙΟ" or perioxi =="Ρίο" or perioxi =="ρίο":

proForou = ektasi \* 800

foros = (proForou \* 24) / 100

metaForou = proForou + foros

print ("Η τιμή του ακινήτου με τον φόρο είναι €" + str(metaForou))

else:

print("Η περιοχή δε βρέθηκε! \n Δοκιμάστε: ΚΕΝΤΡΟ, ΑΓΙΑ ΣΟΦΙΑ, ΡΙΟ")

Δραστηριότητα 14

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει την τιμή ενός παγωτού μηχανής ανάλογα με το βάρος, τη γεύση και το extra περιεχόμενό του.

Γεύση σοκολάτα = 1€/100γρ

Γεύση βανίλια = 0,90€/100γρ

Γεύση φράουλα = 1,10€/100γρ

Αμύγδαλο = 0,20€

Σιρόπι = 0,10€

Φρούτα = 0,50€

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής, if, συγκρίσεις*

*Τρόπος σκέψης:*

*Α) Ορίζουμε 5 μεταβλητές (gefsi, varos, siropi, amygdalo, frouta), τις τιμές των οποίων εισάγει ο χρήστης. Στο κείμενο που εμφανίζεται στον χρήστη γίνεται σαφές πως πρέπει να είναι διατυπωμένες οι απαντήσεις του, ώστε να μπορούν να χρησιμοποιηθούν παρακάτω.*

*Ορίζουμε μια μεταβλητή (timi) της οποίας η τιμή είναι 0, για να ορίσουμε ότι η αρίθμηση ξεκινάει από το 0 και προστίθενται, ανάλογα με τις επιλογές του χρήστη, το κατάλληλο ποσό.*

*Β) Ορίζονται 4 συνθήκες σχετικά με το τι γεύση έχει επιλέξει ο χρήστης (* if gefsi == 1 *) και πόσο αυτή κοστίζει ανάλογα με το βάρος του παγωτού (* timi = (varos \* 1)/100 *), καθώς και αν ο χρήστης έχει επιλέξει να προσθέσει στο παγωτό σιρόπι (* if siropi == "ΝΑΙ" *), αμύγδαλο (* if amygdalo == "ΝΑΙ" *) και φρούτα (* if frouta == "ΝΑΙ" *). Ανάλογα αναπροσδιορίζεται κάθε κάθε φορά η τιμή.*

*Γ) Εμφανίζεται στην οθόνη του υπολογιστή η συνολική τιμή του παγωτού (* print("Το παγωτό σου κοστίζει €" + str(timi)) *).*

*Ενδεικτικός τρόπος επίλυσης:*

gefsi = int(input ("Τι γεύση θέλετε; \n Γράψτε 1 για σοκολάτα \n Γράψτε 2 για βανίλια \n Γράψτε 3 για φράουλα \nΕπιλέξτε:"))

varos = float(input("Πόσα γραμμάρια ζυγίζει το παγωτό;\nΑπαντήστε:"))

siropi = str(input('Έχετε σιρόπι; ΝΑΙ ή ΟΧΙ\nΑπαντήστε:'))

amygdalo = str(input('Έχετε αμύγδαλο; ΝΑΙ ή ΟΧΙ\nΑπαντήστε:'))

frouta = str(input('Έχετε φρούτα; ΝΑΙ ή ΟΧΙ\nΑπαντήστε:'))

timi = 0

if gefsi == 1:

timi = (varos \* 1)/100

elif gefsi == 2:

timi = (varos \* 0.90)/100

elif gefsi == 3:

timi = (varos \* 1.10)/100

if siropi == "ΝΑΙ":

timi = timi + 0.10

if amygdalo == "ΝΑΙ":

timi = timi + 0.20

if frouta == "ΝΑΙ":

timi = timi + 0.50

print("Το παγωτό σου κοστίζει €" + str(timi))

Δραστηριότητα 15

Να φτιάξετε ένα πρόγραμμα που να υπολογίζει το ποσό του φιλοδωρήματος που αντιστοιχεί σε κάθε σερβιτόρο ενός εστιατορίου. Κάθε βασικός σερβιτόρος έχει έναν βοηθό, ο οποίος δικαιούται το 1/3 των φιλοδωρημάτων.

*Θα χρειαστείτε: input, print, πράξεις, ορισμό τύπου μεταβλητής, if, συγκρίσεις*

*Τρόπος σκέψης:*

*Α) Ορίζουμε μια μεταβλητή (tips) τύπου float, τιμή της οποίας είναι το συνολικό ποσό των φιλοδωρημάτων, και δύο μεταβλητές (servitoroiA, servitoroiB) τύπου int, τιμή των οποίων είναι ο αριθμός των βασικών και των βοηθών σερβιτόρων. Τις τιμές των μεταβλητών τις εισάγει ο χρήστης.*

*Β) Ορίζουμε μια μεταβλητή (* poso\_ana\_2 *), τιμή της οποίας είναι το αποτέλεσμα της πράξης που διαιρεί τα tips με βάση τον αριθμό των βασικών σερβιτόρων (* poso\_ana\_2 = tips / servitoroiA *) και εμφανίζεται το ποσό που αναλογεί σε κάθε ομάδα (βασικός και βοηθός του) σερβιτόρων (* print("Το ποσό των φιλοδωρημάτων είναι", poso\_ana\_2, "ευρώ για κάθε ομάδα σερβιτόρων") *)*

*Γ) Ορίζουμε δύο μεταβλητές όπου ορίζεται το ποσό που αντιστοιχεί σε κάθε βασικό (* tips\_A = (poso\_ana\_2 / 3) \* 2 *) και σε κάθε βοηθό (* tips\_B = poso\_ana\_2 % tips\_A *) σερβιτόρο εμφανίζεται το αποτέλεσμα (* print("Βασικοί σερβιτόροι: €" + str(tips\_A) +

" Βοηθοί σερβιτόροι: €" + str(tips\_B)) *).*

*Ενδεικτικός τρόπος επίλυσης:*

tips = float(input("Ποιό είναι το συνολικό ποσό των φιλοδωρημάτων;"))

servitoroiA = int(input("Πόσοι είναι οι βασικοί σερβιτόροι;"))

servitoroiB = int(input("Πόσοι είναι οι βοηθοί σερβιτόροι;"))

poso\_ana\_2 = tips / servitoroiA

print("Το ποσό των φιλοδωρημάτων είναι", poso\_ana\_2, "ευρώ για κάθε ομάδα σερβιτόρων")

tips\_A = (poso\_ana\_2 / 3) \* 2

tips\_B = poso\_ana\_2 % tips\_A

print("Βασικοί σερβιτόροι: €" + str(tips\_A) +

" Βοηθοί σερβιτόροι: €" + str(tips\_B))

# Ενδεικτικά φύλλα εργασίας για τους μαθητές

Τα παρακάτω φύλλα εργασίας προτείνονται ώστε οι μαθητές να μάθουν τη γλώσσα προγραμματισμού Python και να κατανοήσουν τον τρόπο που λειτουργεί μια κειμενική γλώσσα προγραμματισμού. Είναι σκόπιμο να δοθούν στους μαθητές πριν τις [δραστηριότητες](#_Δραστηριότητες) που προτείνονται παραπάνω ώστε να αναπτύξουν αρχικά ευχέρεια ως προς τη χρήση της γλώσσας και αργότερα να έχουν τη δυνατότητα να επικεντρωθούν στην ανάπτυξη αλγορίθμου με βάση τις [δραστηριότητες](#_Δραστηριότητες) και όχι στη γλώσσα αυτή καθαυτή.

Αναλυτικά για τα φύλλα εργασίας:

|  |  |
| --- | --- |
| Φύλλο εργασίας 1 | |
| Διάρκεια:  2 διδακτικές ώρες | Στόχοι:  Οι μαθητές αναμένεται να γνωρίσουν:   1. Τι είναι οι γλώσσες προγραμματισμού 2. Πως να εμφανίζουν περιεχόμενο στην οθόνη του Η/Υ 3. Πως να κάνουν πράξεις με αριθμούς και με κείμενα 4. Τι είναι η μεταβλητή και πως χρησιμοποιείται στον προγραμματισμό |
| Φύλλο εργασίας 2 | |
| Διάρκεια:  2 διδακτικές ώρες | Στόχοι:  Οι μαθητές αναμένεται να γνωρίσουν:   1. Πως να δημιουργούν έναν αλγόριθμο με βάση διαφορετικά κάθε φορά δεδομένα (εισαγωγή δεδομένων από το χρήστη) 2. Τα είδη μεταβλητών και πως χρησιμοποιούνται |
| Φύλλο εργασίας 3 | |
| Διάρκεια:  1 διδακτική ώρα | Στόχοι:  Οι μαθητές αναμένεται να γνωρίσουν:   1. Πως να δημιουργούν έναν αλγόριθμο με βάση έναν έλεγχο συνθηκών |
| Φύλλο εργασίας 4 | |
| Διάρκεια:  2 διδακτικές ώρες | Στόχοι  Οι μαθητές αναμένεται να γνωρίσουν:   1. Πως να δημιουργούν έναν αλγόριθμο με σύγκριση συνθηκών 2. Πως να ελέγχουν συνθήκες ως αληθείς ή ψευδείς |
| Φύλλο εργασίας 5 |  |
| Διάρκεια:  1 διδακτική ώρα | Στόχοι:  Οι μαθητές αναμένεται να γνωρίσουν:   1. Πως να δημιουργούν έναν αλγόριθμο που θα επαναλαμβάνεται για όσο ισχύει μια συνθήκη |
| Φύλλο εργασίας 6 & 7 |  |
| Διάρκεια:  2 διδακτικές ώρες | Στόχοι:  Οι μαθητές αναμένεται να γνωρίσουν:   1. Πως να δημιουργούν μια λίστα 2. Πως να αναζητούν αντικείμενα μέσα σε μία λίστα 3. Πως να χρησιμοποιούν λειτουργίες μέσα σε λίστες: εμφάνιση ενός ή όλων των αντικειμένων μιας λίστας, εισαγωγή αντικειμένων σε λίστα, άθροισμα των αντικειμένων μιας λίστας |
| Φύλλο εργασίας 8 & 9 | |
| Διάρκεια:  1 διδακτική ώρα | Στόχοι:  Οι μαθητές αναμένεται να γνωρίσουν:   1. Πως να μετατρέπουν μια μεταβλητή σε λίστα 2. Πως να δημιουργούν ένα εύρος αριθμών 3. Πως να χρησιμοποιούν ένα εύρος αριθμών για να δημιουργήσουν μια δομή επανάληψης |
| Φύλλο εργασίας 10 | |
| Διάρκεια:  1 διδακτική ώρα | Στόχοι:  Οι μαθητές αναμένεται να γνωρίσουν:   1. Την έννοια της αποδοτικότητας του αλγορίθμου 2. Πως να σχεδιάζουν έναν αποδοτικό αλγόριθμο |
| Φύλλο εργασίας 11 | |
| Διάρκεια:  1 διδακτική ώρα | Στόχοι:  Οι μαθητές αναμένεται να γνωρίσουν:   1. Πως να εισάγουν και να χρησιμοποιούν μια βιβλιοθήκη (εδώ την βιβλιοθήκη «turtle») |

**Φύλλο εργασίας 1**

***Τι είναι γλώσσα προγραμματισμού;***

Με μια γλώσσα προγραμματισμού φτιάχνουμε προγράμματα. Πρόγραμμα είναι μια σειρά από οδηγίες που δίνουμε στον επεξεργαστή του υπολογιστή, για να λύσει ένα πρόβλημα. Το πρόγραμμα μπορούμε να το αποθηκεύσουμε για να μην το ξαναγράφουμε κάθε φορά.

Κάθε γλώσσα προγραμματισμού έχει δικό της λεξιλόγιο και δική της σύνταξη. Σε αντίθεση με τις γλώσσες που μιλάμε, η σύνταξη των γλωσσών προγραμματισμού είναι πάρα πολύ αυστηρή, δηλαδή κάθε λέξη χρησιμοποιείται με συγκεκριμένο τρόπο - και αν δεν τη χρησιμοποιήσουμε έτσι το πρόγραμμα δεν λειτουργεί.

Η Python είναι μία από τις πιο γνωστές γλώσσες προγραμματισμού και χρησιμοποιείται για να φτιάχνουμε προγράμματα που λειτουργούν (τρέχουν) σε υπολογιστές, smartphones, ιστοσελίδες και διάφορες ηλεκτρονικές συσκευές.

***Το πρώτο μας πρόγραμμα σε Python...***

Στην Python χρησιμοποιούμε πολύ συχνά την εντολή print.

Μετά το print βάζουμε δύο παρενθέσεις και καμιά φορά δύο εισαγωγικά (βλ. παρακάτω πίνακα):



***Γράψτε στον υπολογιστή το κείμενο που φαίνεται στον πίνακα 1.1 και καταγράψτε το αποτέλεσμά του.***

Πίνακας 1. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| print(“Hello world”) |  |



***Μπορείτε να περιγράψετε τι κάνει η εντολή print;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



***Έχετε ξαναδεί παρόμοια εντολή σε κάποιο άλλο πρόγραμμα Η/Υ;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***Φτιάξτε ένα δικό σας παράδειγμα, χρησιμοποιώντας την εντολή print.***

Πίνακας 1. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| \_\_\_\_\_\_\_\_(“\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”\_\_\_ |  |

***Αριθμοί και πράξεις***

Για να κάνουμε πράξεις στην Python χρησιμοποιούμε τα σύμβολα: Πρόσθεση **+**, Αφαίρεση **-**, Πολλαπλασιασμό **\***, Διαίρεση **/**.

Πίνακας 1. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| 5 + 5 + 3 | 13 |
| 5 \* 3 | 15 |
| 10 / 2 | 5 |
| 10 \_\_\_ \_\_\_ | 6 |
| 64 / \_\_\_ | 32 |

Αν γράψουμε τις παραπάνω πράξεις, ο υπολογιστής θα τις εκτελέσει, όμως, δε θα μας δείξει το αποτέλεσμα στην οθόνη, γιατί δεν του το ζητήσαμε. Για να δείξουμε το αποτέλεσμα της πράξης στην οθόνη, πρέπει να γράψουμε:

Πίνακας 1. 4

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| \_\_\_\_\_\_\_\_(5 + 5) | 10 |



***Ποια διαφορά παρατηρείτε στη σύνταξη της εντολής; Για ποιο λόγο νομίζετε ότι γίνεται αυτό;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



***Πώς θα υπολογίσουμε και θα εμφανίσουμε στην οθόνη το αποτέλεσμα της πράξης 256 \* 64;***

Πίνακας 1. 5

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| \_\_\_\_\_\_\_\_( \_\_\_ \* \_\_\_\_ ) |  |

***Χαρακτήρες και κείμενα***

Όπως καταλάβατε, όταν στο πρόγραμμά μας δεν θέλουμε να κάνουμε πράξεις (π.χ. πρόσθεση, αφαίρεση, πολ/μός, διαίρεση) χρησιμοποιούμε τα **“ ”**. Για παράδειγμα δεν μπορούμε να κάνουμε πράξεις με μια πρόταση, οπότε μια πρόταση πρέπει να είναι μέσα σε **“ ”**.

Πίνακας 1. 6

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| print(“Hello”) | Hello |
| print(“3 + 5 kanei 8”) | 3 + 5 kanei 8 |
| print(“3 + 5”) | \_\_\_\_\_\_\_\_\_\_\_\_ |
| ***όμως...*** | |
| print(3 + 5) | 8 |
| print(10 + 10) | \_\_\_\_\_\_ |



***Δείξτε στην οθόνη τον αριθμό 1000 με όποιον τρόπο θέλετε:***

Πίνακας 1. 7

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| \_\_\_\_\_\_\_\_( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ) | 1000 |

***Μεταβλητές***

Στον προγραμματισμό πολύ συχνά χρησιμοποιούμε τον όρο «μεταβλητή». Η μεταβλητή είναι ένας χώρος στη μνήμη του υπολογιστή που έχει ένα όνομα και σε αυτόν αποθηκεύεται μια πληροφορία (κείμενο ή αριθμό), που λέγεται τιμή.

**onoma** = “Μπομπ Σφουγγαράκης Τετραγωνοπαντελονής”

Στο παραπάνω παράδειγμα η λέξη **onoma** είναι η μεταβλητή, το σύμβολο **=** λέει στον υπολογιστή τι πληροφορία θα αποθηκεύσουμε στη μεταβλητή και η φράση **Μπομπ Σφουγγαράκης Τετραγωνοπαντελονής**είναι η τιμή της μεταβλητής.

*Μερικά παραδείγματα:*

**zwo** = “σκύλος”

Στο παραπάνω παράδειγμα φτιάξαμε μια μεταβλητή που ονομάζεται zwo και έχει την πληροφορία (τιμή) “σκύλος”.

**varos** = 30 + 2

Αυτή η μεταβλητή ονομάζεται **varos** και έχει την πληροφορία (τιμή) 32 (που βρέθηκε επειδή προστίθενται το 30 και το 2).



***Βλέπουμε ότι παραπάνω γράψαμε “σκύλος” ΑΛΛΑ γράψαμε 30 + 2 χωρίς “”. Γιατί έγινε αυτό;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



***Φτιάξτε μια μεταβλητή με τιμή την ηλικία σας:***

Πίνακας 1. 8

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| \_\_\_\_\_\_\_\_ = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |

Τα ονόματα των μεταβλητών τα διαλέγουμε εμείς.

Μπορούμε να ενώσουμε όσες μεταβλητές θέλουμε, π.χ.:

**onoma** = “Nefeli”

**eponimo** = “Kostakopoulou”

Πίνακας 1. 9

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| onoma = “Nefeli”  eponimo = “Kostakopoulou”  print(onoma)  print(eponimo)  print(onoma + eponimo)  print(“onoma”) | Nefeli  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  NefeliKostakopoulou  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Προσοχή: Το **=** στον προγραμματισμό χρησιμοποιείται **διαφορετικά** απ’ ό,τι στα μαθηματικά

Πράξεις με μεταβλητές για να βρούμε το βάρος δύο παιδιών:

Πίνακας 1. 10

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| paidi1 = 35  paidi2 = 35  paidi3 = 40  print(paidi1 + paidi2)  print(paidi1 + paidi3) | 70  \_\_\_\_\_ |



***Τι αποτέλεσμα πιστεύετε ότι έχει η τελευταία γραμμή του παραπάνω κώδικα; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



***Φτιάξτε μια μεταβλητή με το όνομά σας και γράψτε την κατάλληλη εντολή, ώστε ο υπολογιστής να δείχνει την τιμή της στην οθόνη:***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**Φύλλο εργασίας 2**

***Επίλυση απλών προβλημάτων***

Με αυτά που μάθαμε ως εδώ μπορούμε σχεδόν να αρχίσουμε να λύνουμε απλά προβλήματα μαθηματικών με τη βοήθεια του υπολογιστή.

Πριν ξεκινήσουμε θα δούμε μια ακόμη εντολή στην Python που μας δίνει τη δυνατότητα να δίνουμε στον υπολογιστή τα δεδομένα που πρέπει να χρησιμοποιήσει για να λύσει ένα πρόβλημα. Η εντολή **input** κάνει ακριβώς αυτό.

ΠΡΟΣΟΧΗ: Στην Python κάθε μεταβλητή έχει έναν «τύπο», ανάλογα με τα δεδομένα που έχουμε αποθηκεύσει σε αυτή. Οι πιο σημαντικοί τύποι μεταβλητών είναι **int** (ακέραιος αριθμός), **float** (δεκαδικός αριθμός) και **str** (κείμενο). Όταν χρησιμοποιούμε την εντολή input, η Python θεωρεί ότι θα εισάγουμε «κείμενο», δηλαδή μια μεταβλητή που τα δεδομένα της προέρχονται από input θα είναι αυτομάτως τύπου str (string). Για να εισάγουμε με το input «αριθμούς» πρέπει να χρησιμοποιήσουμε τις εντολές int() ή float(), π.χ. **x = int(input(“Γράψτε έναν ακέραιο αριθμό:”))**

Πίνακας 2. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| x = input(“Ζώο: ”) | Ζώο: |

Τώρα ο υπολογιστής θα περιμένει να γράψουμε ένα «ζώο» για να συνεχίσει.

**Γράφουμε «γάτα».**

Πίνακας 2. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| x = input(“Ζώο: ”)  print(x) | Ζώο: γάτα  γάτα |

ΠΡΟΣΟΧΗ: Στο παραπάνω παράδειγμα ο υπολογιστής μας ζήτησε να του πούμε ένα «ζώο» και αποθήκευσε την πληροφορία που του δώσαμε στη μεταβλητή x. Στη συνέχεια, έδειξε την τιμή της μεταβλητής x στην οθόνη.



***Πείτε στον υπολογιστή να ζητήσει το όνομά σας και μετά να το δείξει στην οθόνη:*** 

Πίνακας 2. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| onoma = input(“\_\_\_\_\_\_\_\_\_\_\_: ”)  print(onoma) | Όνομα: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |



***Αν γράψουμε*** x = int(input(“Γράψτε κάτι εδώ:”)) ***τι τύπο μεταβλητής (κείμενο, ακέραιο, δεκαδικό) περιμένει ο Η/Υ να γράψουμε και γιατί;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



***Φτιάξτε μια μεταβλητή που να βάζετε την ηλικίας σας (αριθμό) χρησιμοποιώντας το input και μετά να τη δείχνετε στην οθόνη:***

Πίνακας 2. 4

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| x = \_\_\_\_\_(input(“Ηλικία: ”))  ­­­­\_\_\_\_\_\_\_\_\_\_(x) | \_\_\_\_\_\_\_\_\_:  \_\_\_\_ |



***Φτιάξτε μια μεταβλητή που να βάζετε το βάρος σας (αριθμό) χρησιμοποιώντας το input και μετά να το δείχνετε στην οθόνη:***

Πίνακας 2. 5

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  |  |



*Πρόβλημα 1*

Ας φτιάξουμε ένα πρόγραμμα που να υπολογίζει το Μέσο Όρο (ΜΟ) δύο ακέραιων αριθμών (που κάθε φορά θα λέμε εμείς στον υπολογιστή ποιοι θέλουμε να είναι).

Με αυτά που έχουμε μάθει ως τώρα:

***Πώς θα πούμε στον υπολογιστή να μας ζητήσει να του δώσουμε 2 αριθμούς και να τους θυμάται;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



***Τι μαθηματικές πράξεις πρέπει να κάνει ο Η/Υ για να βρει τον ΜΟ των αριθμών;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



***Ποιες εντολές θα χρησιμοποιήσουμε για να:***

*Δηλώσουμε ακέραιους αριθμούς: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Εισάγουμε τους αριθμούς αυτούς: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Εμφανίσουμε το ΜΟ στην οθόνη: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

***Ο παρακάτω κώδικας βρίσκει το ΜΟ 2 αριθμών:***

x = int(**input**("αριθμός 1:"))

y = int(**input**("αριθμός 2:"))

**print**("Ο μέσος όρος των 2 αριθμών είναι:" , (x + y)/2)

***Πως μπορούμε να αλλάξουμε τον παραπάνω κώδικα ώστε να υπολογίζεται ο Μέσος Όρος τριών διαφορετικών κάθε φορά αριθμών;***

Πίνακας 2. 6

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| x = \_\_\_\_(\_\_\_\_\_\_("αριθμός 1:"))  \_\_ = \_\_\_\_(\_\_\_\_\_\_ ("αριθμός 2:"))  \_\_ = \_\_\_\_(\_\_\_\_\_\_ ("αριθμός 3:"))  **print**("Ο μέσος όρος των 3 αριθμών είναι:" , (x + \_\_ + \_\_)/\_\_) |  |

***Περιγράψτε τι κάνουν οι 3 πρώτες και η 4η γραμμή του κώδικα στον πίνακα 2.6:***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



*Πρόβλημα 2*

Ας φτιάξουμε ένα πρόγραμμα που να μετατρέπει σε δέκατα και εκατοστά την απόσταση που εισάγει ο χρήστης σε μέτρα.

Με αυτά που έχουμε μάθει ως τώρα:



***Πώς θα πούμε στον υπολογιστή να μας ζητήσει να του δώσουμε 1 αριθμό (δεκαδικό) και να τον θυμάται;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



***Τι μαθηματικές πράξεις πρέπει να κάνει ο Η/Υ για να μετατρέψει τα μέτρα σε δέκατα και τι για εκατοστά;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***



***Ποιες εντολές θα χρησιμοποιήσουμε για να:***

*Εισάγουμε δεκαδικούς αριθμούς: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Εισάγουμε τον αριθμό αυτόν: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Εμφανίσουμε τις μετατροπές στην οθόνη: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

***Ο παρακάτω κώδικας μετατρέπει τα μέτρα σε δέκατα και εκατοστά:***

x = float(**input** ("Εισάγετε σε μέτρα την απόσταση που θέλετε να μετατρέψετε:"))

y = float(x \* 10)

**print**(y , "δέκατα")

z = float(x \* 100)

**print**(z , " εκατοστά")

***Πως μπορούμε να αλλάξουμε τον παραπάνω κώδικα ώστε να μετατρέπεται η απόσταση σε χιλιοστά και χιλιόμετρα;***

Πίνακας 2. 7

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| \_\_\_ = ­­­­\_\_\_\_\_\_(\_\_\_\_\_\_("Εισάγετε σε μέτρα την απόσταση:"))  \_\_\_ = \_\_\_\_\_\_ (\_\_\_ \* 1000)  **print** (\_\_\_ , "χιλιοστά")  \_\_\_ = float (\_\_\_ \_\_ \_\_\_\_)  \_\_\_\_\_(\_\_\_ , "χιλιόμετρα") |  |

**Φύλλο εργασίας 3**

***Έλεγχος συνθηκών – εντολή if,else***

Χρησιμοποιούμε τη εντολή **εάν** (**if**) για να πούμε στον υπολογιστή να κάνει κάτι μόνο αν ισχύει κάτι άλλο.

*Παράδειγμα 1:*

if 3 < 10:

**print**('Hello')

Ο υπολογιστής θα εμφανίσει τη λέξη «Hello» **γιατί ισχύει ότι 3 < 10**.



***Τι κάνει η 1η και τι η 2η γραμμή του κώδικα στο παράδειγμα 1;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

*Παράδειγμα 2:*

if 64 / 2 > 128:

**print**('Hello')



***Τι νομίζετε ότι θα κάνει ο Η/Υ αν γράψουμε τον κώδικα από το παράδειγμα 2; Γιατί;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

*Παράδειγμα 3:*

kostos = 15

if kostos >= 20:

**print**('Είναι ακριβό')

Ο υπολογιστής **δε** θα εμφανίσει τη φράση «Είναι ακριβό» γιατί το 15 **δεν είναι >=** του 20.



***Πώς πρέπει να αλλάξουμε τον παραπάνω κώδικα ώστε ο Η/Υ να εμφανίζει το μήνυμα «Είναι ακριβό»;***

Πίνακας 3. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  |  |

Την εντολή **if** μπορεί να ακολουθεί η εντολή **else**, η οποία περιλαμβάνει κώδικα που πραγματοποιείται αν η συνθήκη αποδειχθεί ψευδής (**εάν…αλλιώς…**).

*Παράδειγμα 4:*

kostos = 15

if kostos >= 20:

**print**('Είναι ακριβό')

else:

**print**('Είναι φθηνό')

Ο υπολογιστής θα εμφανίσει τη φράση «Είναι φθηνό», που «βρίσκεται μέσα» στο **else**, γιατί το 15 **δεν είναι >=** του 20.



***Τώρα φτιάξτε ένα δικό σας πρόβλημα που να ελέγχει την ηλικία που θα πούμε στον Η/Υ και θα μας εμφανίζει «είσαι ενήλικος» ή «είσαι ανήλικος»:***

Πίνακας 3. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  |  |

Ας φτιάξουμε ένα πρόγραμμα όπου θα εισάγουμε την ηλικία μας και θα υπολογίζεται πόσα χρόνια χρειάζονται ακόμα για να ενηλικιωθούμε.

x = **input**("Εισάγετε ηλικία:")

**print**(18 - x)



*Πρόβλημα*

*Στηριζόμενοι στα παραπάνω, ας φτιάξουμε ένα πρόγραμμα όπου ο χρήστης θα εισάγει μια ηλικία και θα ελέγχεται αν είναι ενήλικος και σε περίπτωση που είναι ανήλικος να υπολογίζεται σε πόσα χρόνια θα ενηλικιωθεί. Το αποτέλεσμα να εμφανίζεται στην οθόνη.*

* Ο υπολογιστής θα κάνει πάντα το ίδιο πράγμα όποια κι αν είναι η ηλικία που θα εισάγει ο χρήστης; Γιατί ή γιατί όχι;

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

* Πως ο Η/Υ θα ξεχωρίσει αν ο χρήστης είναι ενήλικος ή ανήλικος; Ποιες εντολές θα χρησιμοποιήσουμε;

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

* Πως θα υπολογίσουμε πόσα χρόνια χρειάζονται μέχρι την ενηλικίωση;

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

* Γράψε τον κώδικα:

Πίνακας 3. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  |  |

**Φύλλο εργασίας 4**

***Τύπος μεταβλητής Boolean και Άλγεβρα Boole***

Στις περισσότερες γλώσσες προγραμματισμού υπάρχει ο τύπος μεταβλητής Boolean, ο οποίος μπορεί να πάρει μόνο δύο τιμές True (Αληθές) ή False (Ψευδές).



***Δοκιμάστε στον Η/Υ τον παρακάτω κώδικα και γράψτε το αποτέλεσμα:***

Πίνακας 4. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| print(2 == 8)  print(2 != 8)  print(8 != 8)  print(2 > 8)  print(2 < 8) |  |



***Σε ποιες περιπτώσεις το αποτέλεσμα ήταν «True» και γιατί νομίζετε ότι έγινε αυτό;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Τι ελέγχουν, λοιπόν, τα παρακάτω σύμβολα;***

== \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

!= \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

< \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Δοκιμάστε τον παρακάτω κώδικα:***

a = int(input("1ος αριθμός:"))

b = int(input("2ος αριθμός:"))

print("Οι αριθμοί είναι ίσοι:", a==b)



***Τι κάνουν οι δύο πρώτες γραμμές;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Πότε το αποτέλεσμα είναι «True» και πότε «False»;***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



***Τι χρησιμότητα έχει το κόμμα (,) μέσα στην εντολή print;***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Μέχρι τώρα είδαμε πώς μπορούμε να ελέγχουμε αν μία συνθήκη είναι αληθής ή ψευδής.

Πολλές φορές χρειάζεται να ελέγξουμε παραπάνω από μία συνθήκες ταυτόχρονα. Για να το κάνουμε αυτό χρησιμοποιούμε τις λέξεις and, or και not.



***Δοκιμάστε στον Η/Υ τον κώδικα του πίνακα 4.2 και γράψτε το αποτέλεσμα:***

Πίνακας 4. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| print(2 < 8 and 8 == 8)  print(2 > 8 and 8 == 8)  print(8 != 8 or 16 == 16)  print(8 != 8 or 16 != 16)  print(not 2 < 8) |  |



***Σε ποιες περιπτώσεις το αποτέλεσμα ήταν «True» και γιατί νομίζετε ότι έγινε αυτό;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Συνήθως χρησιμοποιούμε την Άλγεβρα Boole μαζί με το if ή/και το else για να συγκρίνουμε μεταβλητές και αναλόγως να αποφασίσουμε τι θα κάνει το πρόγραμμά μας.

Πίνακας 4. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| a = 18  b = 17  c = 18  if a >= 18 and b >= 18 and c >= 18:  print("Είναι όλοι ενήλικοι.") |  |



***Τι νομίζετε ότι κάνει ο κώδικας του πίνακα 4.3; Τι αποτέλεσμα θα εμφανίσει;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Τι πρέπει να αλλάξουμε στον κώδικα του πίνακα 4.3 για να εκτελεστεί η εντολή print; Μπορείτε να βρείτε 2 διαφορετικούς τρόπους;***

Πίνακας 4. 4

|  |  |
| --- | --- |
| 1ος τρόπος | 2ος τρόπος |
|  |  |

**Φύλλο εργασίας 5**

***Δομή επανάληψης while – while loops***

Η δομή επανάληψης while χρησιμοποιείται όταν θέλουμε να γίνεται μια διαδικασία επαναλαμβανόμενα, για όσο ισχύει μια συνθήκη που έχουμε ορίσει.

Πίνακας 5. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| i = 1  while i <= 10:  print(i)  i = i + 1 | 1  2  3  4  5  6  7  8  9  10 |



***Γιατί ο κώδικας του πίνακα 5.1 εμφάνισε όλους τους αριθμούς από το 1 έως το 10;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Τι χρησιμότητα έχει η τελευταία γραμμή του κώδικα; Τι θα συμβεί αν τη σβήσουμε;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Χρησιμοποιώντας τη συνθήκη*** ***ελέγχου if, πώς μπορούμε να αλλάξουμε τον κώδικα του πίνακα 5.1 ώστε να μας δείχνει τους αριθμούς από το 5 έως το 10 (χωρίς να αλλάξουμε την πρώτη γραμμή);***

Πίνακας 5. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  | 5  6  7  8  9  10 |



*Πρόβλημα*

Φτιάξτε ένα πρόγραμμα που να ζητάει από το χρήστη να εισάγει έναν αριθμό και στη συνέχεια να υπολογίζεται η προπαίδειά του

Πίνακας 5. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  |  |

**Φύλλο εργασίας 6**

***Λίστες - lists***

Η λίστα μοιάζει με τη μεταβλητή, με τη διαφορά, όμως, πως η λίστα μπορεί να αποθηκεύσει πολλές τιμές. Για παράδειγμα, μπορούμε να φτιάξουμε μια μεταβλητή zwo που έχει την τιμή "γάτα", όμως, αν θα θέλαμε η μεταβλητή μας να έχει περισσότερες από μία τιμές, θα ορίζαμε μία λίστα:

zwa = ["γάτα", "σκύλος", "κότα"]

Μία λίστα μπορεί να είναι ακόμη και κενή:

lista = []

Κάθε τιμή της λίστας ονομάζεται αντικείμενο και μπορούμε να το χρησιμοποιήσουμε με τον παρακάτω τρόπο. Στις λίστες ξεκινάμε να αριθμούμε τα αντικείμενα ξεκινώντας από το 0.

Πίνακας 6. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| zwa = ["γάτα", "σκύλος", "κότα"]  print(zwa[1])  lista = []  print(lista) | σκύλος  [ ] |

Με βάση τον κώδικα του πίνακα 6.1, συμπληρώστε τον πίνακα 6.2:

Πίνακας 6. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| zwa = ["γάτα", "σκύλος", "κότα"]  print(zwa[ ])  print(zwa[ ])  print( , ) | κότα  γάτα  σκύλος γάτα |

Υπάρχουν πολλές εντολές που μπορούμε να χρησιμοποιήσουμε για τις λίστες.

Πίνακας 6. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| lista = ["γάλα", 8, "σοκολάτα", 16, 3.14]  print(8 in lista)  print("τυρί" in lista)  print(not "τυρί" in lista)  print(not 16 in lista)  print(16 in lista and 32 in lista) | True  False  True  False  False |

***Μπορείτε να εξηγήσετε τι ελέγχουν οι εντολές in και not;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Εξηγείστε γιατί η τελευταία γραμμή έχει αποτέλεσμα False:***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Στο παρακάτω παράδειγμα χρησιμοποιούμε τη λίστα μαζί με το if, κάτι που συνηθίζεται στον προγραμματισμό.

Πίνακας 6. 4

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| numbers = [1, 2, 3, 4, 5]  x = int(input("Εισάγετε αριθμό: "))  if x in numbers:  print("Βρέθηκε") | Εισάγετε αριθμό: 4  Βρέθηκε |

ΠΡΟΣΟΧΗ: Μετά το if ελέγχεται η συνθήκη x in numbers, η οποία είναι True. Δηλαδή η συνθήκη είναι αληθής και γι’ αυτό το λόγο εκτελείται και η εντολή print που είναι μέσα στο if.



*Πρόβλημα*

Φτιάξτε τώρα ένα πρόγραμμα που περιέχει μια λίστα με τους αριθμούς: 8, 16, 32, 64, 128. Ονομάστε τη λίστα όπως θέλετε. Ζητήστε από τον χρήστη να εισάγει έναν ακέραιο αριθμό. Φροντίστε ο υπολογιστής να αποθηκεύσει τον αριθμό. Ελέγξτε αν ο αριθμός υπάρχει στη λίστα. Αν υπάρχει, να εμφανίζεται το μήνυμα «Βρέθηκε», αλλιώς το μήνυμα «Δε βρέθηκε».

Πίνακας 6. 5

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  |  |

**Φύλλο εργασίας 7**

***Λειτουργίες σε λίστες***

Αφού δημιουργήσουμε μία λίστα, μπορούμε να προσθέτουμε ή/και να αλλάζουμε τα αντικείμενά της.

Μπορούμε να εμφανίσουμε όλα τα αντικείμενα της λίστα μαζί ή καθένα ξεχωριστά, όπως φαίνεται παρακάτω:

Πίνακας 7. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| zwa = ["γάτα", "σκύλος", "κότα"]  print(zwa)  print(zwa[0])  print(zwa[1])  print(zwa[2]) | ['γάτα', 'σκύλος', 'κότα']  γάτα  σκύλος  κότα |



***Παρατηρήστε το παρακάτω:***

Πίνακας 7. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| zwa = ["γάτα", "σκύλος", "κότα"]  print(zwa)  zwa.append("καναρίνι")  print(zwa)  zwa.append(256)  print(zwa) | ['γάτα', 'σκύλος', 'κότα']  ['γάτα', 'σκύλος', 'κότα', 'καναρίνι']  ['γάτα', 'σκύλος', 'κότα', 'καναρίνι', 256] |



***Τι νομίζετε ότι κάνει το .append;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Πώς συντάσσεται το .append;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Γιατί γράψαμε το 256 χωρίς εισαγωγικά;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Παρατηρήστε το παρακάτω:***

Πίνακας 7. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| zwa = ["γάτα", "σκύλος", "κότα"]  print(zwa)  zwa.insert(1, "ποντίκι")  print(zwa)  zwa.insert(0, "σπουργίτι")  print(zwa) | ['γάτα', 'σκύλος', 'κότα']  ['γάτα', 'ποντίκι', 'σκύλος', 'κότα']  ['σπουργίτι', 'γάτα', 'ποντίκι', 'σκύλος', 'κότα'] |



***Μπορείτε να εξηγήσετε τι κάνει το .insert;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Πώς συντάσσεται το .insert;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Τι διαφορά έχει το .append με το .insert;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Για να αλλάξουμε ένα αντικείμενο μιας λίστας κάνουμε ό,τι ακριβώς και με την τιμή μιας μεταβλητής:

Πίνακας 7. 4

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| zwa = ["γάτα", "σκύλος", "κότα"]  print(zwa)  zwa[1] = "ποντίκι"  print(zwa) | ['γάτα', 'σκύλος', 'κότα']  ['γάτα', 'ποντίκι', 'κότα'] |

*Πρόβλημα 1*

Με βάση όσα έχουμε μάθει ως τώρα, φτιάξτε ένα πρόγραμμα όπου ο χρήστης θα εισάγει 2 αριθμούς. Οι αριθμοί θα μπαίνουν σε μία λίστα, η οποία έχει ήδη τους αριθμούς 11 και 12. Αν ένας από τους αριθμούς της λίστας είναι μεγαλύτερος από το 10, να αντικαθίσταται με το 1. Στο τέλος να εμφανίζεται όλη η λίστα.

Πίνακας 7. 5

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  | 1ος αριθμός:  2ος αριθμός:  [ , , , ] |

Μια ακόμη εντολή της Python, που είναι ιδιαίτερα χρήσιμη στις λίστες, είναι η len(). Η εντολή αυτή μάς δίνει το σύνολο των χαρακτήρων μιας μεταβλητής τύπου string ή το σύνολο των αντικειμένων μιας λίστας.

Πίνακας 7. 6

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| x = "nefeli"  y = len(x)  print(y)  zwa = ["γάτα", "σκύλος", "κότα"]  print(len(zwa)) | 6  3 |



***Από πού προκύπτουν τα αποτελέσματα 6 και 3 του παραπάνω κώδικα;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



*Πρόβλημα 2*

Φτιάξτε ένα πρόγραμμα που ο χρήστης θα εισάγει σε μία λίστα 5 ονόματα. Να υπάρχει έλεγχος ώστε να μην μπορεί να μπει στη λίστα 2 φορές το ίδιο χρώμα. Μόλις τα αντικείμενα της λίστας γίνουν 5, να εμφανίζονται στην οθόνη. Να χρησιμοποιήσετε την εντολή len().

Πίνακας 7. 7

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  |  |

**Φύλλο εργασίας 8**

***Εντολές list() και range()***

Η εντολή list() στην Python μετατρέπει μια μεταβλητή σε λίστα.

Πίνακας 8. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| x = "Marina"  y = list(x)  print(y)  print(y[2]) | ['M', 'a', 'r', 'i', 'n', 'a']  r |

Η εντολή range() δημιουργεί ένα εύρος αριθμών, το οποίο για να το αξιοποιήσουμε πρέπει να το βάλουμε μέσα σε μία λίστα με τη εντολή list(). Βασική της σύνταξη είναι με 2 τρόπους, όπως φαίνεται στον παρακάτω πίνακα.

Πίνακας 8. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| x = list(range(10))  y = list(range(5,10))  print(x)  print(y) | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  [5, 6, 7, 8, 9] |



**Από ποιον αριθμό ξεκινάει την αρίθμηση η εντολή** list(range(10));

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Φτιάξτε ένα πρόγραμμα που να βάζει σε μία λίστα αριθμούς από το 1 έως το 1000 και να την εμφανίζει:**

Πίνακας 8. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  | [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, … , 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000] |

**Φύλλο εργασίας 9**

***Δομή επανάληψης for***

Παρόμοια με τη δομή επανάληψης while, το for μάς επιτρέπει να κάνουμε επανάληψη μιας σειράς εντολών, για έναν ορισμένο αριθμό επαναλήψεων, πράγμα που είναι και η διαφορά του από το while.

Χρησιμοποιείται μαζί με την εντολή range().

Πίνακας 9. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| for i in range(5):  print("hello!") | hello!  hello!  hello!  hello!  hello! |

**Παρατηρήστε τον πίνακα:**

Πίνακας 9. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| for i in range(3,5):  print("hello!") |  |



**Τι θα εμφανίσει ο κώδικάς του;**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Πρόβλημα

Φτιάξτε ένα πρόγραμμα που να ζητάει από τον χρήστη να γράψει το όνομά του και στη συνέχεια να εμφανίζει κάθε φορά σε ξεχωριστή γραμμή ένα-ένα τα γράμματα του ονόματός του.

*Μπορείτε να χρησιμοποιήσετε τις εντολές len(), list()και τη δομή επανάληψης for.*

Πίνακας 9. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  |  |

**Φύλλο εργασίας 10**

***Επιλέγοντας την καταλληλότερη λύση (έλεγχος σχεδιασμού και αποδοτικότητας αλγορίθμου)***

Ένα πρόβλημα μπορεί να λύνεται με πολλούς διαφορετικούς τρόπους, όμως πολλές φορές κάποιες λύσεις είναι «καταλληλότερες» από κάποιες άλλες.

Παρατηρήστε τον παρακάτω κώδικα στους δύο πίνακες:

Πίνακας 10. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| x = int(input("αριθμός: "))  if x <= 10:  if x == 2 or x == 4 or x == 6 or x == 8 or x == 10:  print("Ο αριθμός είναι ζυγός.")  else:  print("Ο αριθμός είναι μονός.")  else:  print("Ο αριθμός είναι μεγαλύτερος του 10.") |  |

Πίνακας 10. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| x = int(input("αριθμός: "))  if x <= 10:  if (x % 2) == 0:  print("Ο αριθμός είναι ζυγός.")  else:  print("Ο αριθμός είναι μονός.")  else:  print("Ο αριθμός είναι μεγαλύτερος του 10.") |  |



***Τι νομίζετε ότι κάνει ο κώδικας των πινάκων 10.1 και 10.2; Λειτουργεί σωστά και στις δύο περιπτώσεις;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Τι διαφορά έχει στον τρόπο λειτουργείας ο κώδικας στους δύο πίνακες;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Ο κώδικας ποιου πίνακα εκφράζει καλύτερα τον ορισμό του «μονού» και «ζυγού» αριθμού;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Ο κώδικας ποιας στήλης του πίνακα 10.3 είναι πιο εύκολο να αλλαχθεί ώστε να υπολογίζει αριθμούς ως το 100; Γιατί;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Πίνακας 10. 3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | Αποτέλεσμα |
| print(1)  print(2)  print(3)  print(4)  print(5) | x = [1,2,3,4,5]  i = 0  while i < 5:  print(x[i])  i = i + 1 | x = 1  while x < 6:  print(x)  x = x + 1 | 1  2  3  4  5 |



***Ο κώδικας ποιας στήλης του πίνακα 10.3 είναι ευκολότερο να κατανοηθεί; Γιατί;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Ο κώδικας ποιας στήλης είναι δυσκολότερο να κατανοηθεί; Γιατί;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Αν έπρεπε να αλλάξετε τον κώδικα ώστε να εμφανίζονται οι αριθμοί από το 1-100, ποια στήλη από τις παραπάνω θα διαλέγατε να τροποποιήσετε; Για ποιο λόγο;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Με αυτά που μάθαμε ως τώρα, γράψτε μερικούς κανόνες που πρέπει να ακολουθούμε ώστε να επιλέγουμε τον καταλληλότερο τρόπο για να λύσουμε ένα πρόβλημα στον Προγραμματισμό:***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Φύλλο εργασίας 11**

***Γραφικά χελώνας***

Τα γραφικά χελώνας (turtle) της Python είναι εμπνευσμένα από τη γλώσσα προγραμματισμού Logo, που ήδη γνωρίζουμε.

Για να τα χρησιμοποιήσουμε στο περιβάλλον της Python πρέπει πρώτα να εισάγουμε τη βιβλιοθήκη της Python που τα περιέχει, χρησιμοποιώντας μια λειτουργία που ονομάζεται import, και να καλέσουμε τις ανάλογες εντολές.

Πίνακας 11. 1

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| import turtle  turtle.forward(200)  turtle.left(120)  turtle.forward(200)  turtle.left(120)  turtle.forward(200) |  |



***Περιγράψτε τι κάνει η κάθε γραμμή κώδικα του πίνακα 10.1:***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Παρατηρήστε τον πίνακα 11.2:**

Πίνακας 11. 2

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| import turtle  x = 100  y = 50  turtle.forward(x)  turtle.penup()  turtle.forward(y)  turtle.pendown()  turtle.forward(x) |  |



***Τι νομίζετε ότι κάνει ο κώδικας αυτός;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Παρατηρήστε τον πίνακα 11.3:**

Πίνακας 11. 3

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
| import turtle as t  for i in range(0, 8):  t.forward(100)  t.left(360/8) |  |

***Γράψτε στον Η/Υ και σχεδιάστε το αποτέλεσμα του κώδικα του πίνακα 11.3. Μπορείτε να καταλάβετε τη λειτουργεία της 2ης γραμμής; Τι νομίζετε ότι κάνει;***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Για να σχεδιάσουμε το τρίγωνο στον πίνακα 11.1 χρειάστηκαν έξι γραμμές κώδικα, ενώ τώρα (πίνακας 11.3) μόλις τέσσερις!; Πώς το καταφέραμε αυτό; Τι κάναμε διαφορετικά; Μπορείτε να φανταστείτε κάποια περίπτωση που να μας χρησιμεύει ιδιαίτερα αυτή η τεχνική;***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Πρόβλημα

Με τη βοήθεια των γραφικών χελώνας, φτιάξτε το βασικό σχέδιο ενός σπιτιού, όπως φαίνεται στον πίνακα 11.4:

Πίνακας 11. 4

|  |  |
| --- | --- |
| Κώδικας | Αποτέλεσμα |
|  |  |

# Βιβλιογραφία

Ackermann, E. (2001), Piaget’s Constructivism, Papert’s Constructionism: What’s the difference?, constructivism: Uses and perspectives in education, volymes 1& 2. Conference proceedings, Geneva: Research Center in Education/ Cahier 8/ September 01. [doi: 10.1.1.132.4253](http://citeseerx.ist.psu.edu/viewdoc/citations?doi=10.1.1.132.4253)

Agarwal, K., Agarwal, A., & Celebi, E. (2008). Python puts a squeeze on java for CS0 and

beyond. J. Comput. Sci. Coll. 23, 6, 49-57.

Baron, D. (1994). *Guide to home language repair.* In Proceedings of National Council of Teachers of English, Urbana, IL.

Bransford, J.D., & Donovan, M.S. (2005). Scientific inquiry and how people learn. In National Research Council (ed), *How students learn: History, mathematics, and science in the classroom* (pp. 397-420). Washington DC: The National Academies Press.

Briggs, J. (2013). *Python for kids: A playful introduction to programming*. No Starch Press, Inc.

Bruner, J. (1966). *Toward a theory of instruction*, Cambridge, MA: Harvard University Press

Bruner, J. (1997). *The Culture of Education*. Cambridge, MA: Harvard University Press.

Buabeng-Andoh, C.  (2012). Factors influencing teachers’ adoption and integration of information and communication technology into teaching: A review of the literature. International Journal of Education and Development using Information and Communication Technology (IJEDICT), τόμος. 8, αριθ. 1, σελ. 136-155. Ανακτήθηκε από: <http://files.eric.ed.gov/fulltext/EJ1084227.pdf>

Carey, S. (1987). *Conceptual Change in Childhood*, Cambridge, MA: MIT Press

Carroll, S., Daughtrey, T. (July 4, 2007). Fundamental Concepts for the Software Quality Engineer. American Society for Quality. pp. 282 et seq. ISBN 978-0-87389-720-4

Cormen, Thomas H. (2013), *Algorithms Unlocked*, MIT Press books.

Crook, C. (1998). Children as computer users: The case of collaborative learning. *Computers and Education, 30*(3/4), 237–247.

Europa (2017). *ICT in Education*. Ανακτήθηκε από: <https://ec.europa.eu/digital-single-market/en/ict-education>

European Commission (2011). *Key Data on Learning and Innovation through ICT at School in Europe 2011*. Education, Audiovisual and Culture Executive Agency. Ανακτήθηκε από: <https://publications.europa.eu/en/publication-detail/-/publication/8f864668-0211-4a40-bc14-65bf1a97b6a8>. doi: 10.2797/61068

Futschek G. (2006) *Algorithmic Thinking: The Key for Understanding Computer Science*. In: Mittermeir R.T. (eds) Informatics Education – The Bridge between Using and Understanding Computers. ISSEP 2006. Lecture Notes in Computer Science, vol 4226. Springer, Berlin, Heidelberg. doi: 10.1007/11915355\_15

Garneli, V., Giannakos, M., Chorianopoulos, K. (2015). *Computing Education in K-12 Schools: A Review of the Literature*, [2015 IEEE Global Engineering Education Conference (EDUCON)](https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7086909). doi: [10.1109/EDUCON.2015.7096023](https://doi.org/10.1109/EDUCON.2015.7096023)

Goldwasser, M., & Letscher, D. (2008). Teaching an object-oriented CS1 -: with Python. SIGCSE Bull. 40, 3 (June 2008), 42-46.

[Goodrich, M.](https://en.wikipedia.org/wiki/Michael_T._Goodrich) & [Tamassia, R](https://en.wikipedia.org/wiki/Roberto_Tamassia). (2002). [*Algorithm Design: Foundations, Analysis, and Internet Examples*](http://ww3.algorithmdesign.net/ch00-front.html). John Wiley & Sons, Inc., [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-0-471-38365-9](https://en.wikipedia.org/wiki/Special:BookSources/978-0-471-38365-9)

Gries, D. (1981). *The Science of Programming*. Springer-Verlag

Guido Van Rossum (1999). Computer Programming for Everybody   
(Revised Proposal). Ανακτήθηκε από: <https://www.python.org/doc/essays/cp4e/>

Guo, P., (2014). Python is now the Most Popular Introductory Teaching Language at Top U.S. Universities. Survey published at the Communications of the ACM blog (CACM blog).

Gurevich, Y. (2000). Sequential Abstract State Machines Capture Sequential Algorithms. ACM Transactions on Computational Logic 1

Heintz, F., Mannila, L., Farnqvist, T. (2016). *A Review of Models for Introducing Computational Thinking, Computer Science and Computing in K-12 Education*, [2016 IEEE Frontiers in Education Conference (FIE)](https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7749394). doi: 10.1109/FIE.2016.7757410

Herken, R. (1995). *The Universal Turing Machine: A Half-Century Survey*. Springer Verlag. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [3-211-82637-8](https://en.wikipedia.org/wiki/Special:BookSources/3-211-82637-8)

Hubweiser, P., Giannakos, M., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., Pal, Y., Jackova, J., Jasute, E. (2015). *A Global Snapshot of Computer Science Education in K-12 Schools* In Proceedings of the 2015 ITiCSE Working Group Reports (ITICSE-WGR '15), At Vilnius, Lithuania. doi: 10.1145/2858796.2858799

Hunter, B. (1983), *My students use computers*. Reston, VA: Reston Publishing.

Jonassen, D. H. (2003). *Using cognitive tools to represent problems*. Journal of Research on Technology in Education, 35(3), 362-380. doi: 10.1080/15391523.2003.10782391

[Jones](https://en.wikipedia.org/wiki/Capers_Jones" \o "Capers Jones), C. & Bonsignour, O. (2011). *The Economics of Software Quality*. Addison-Wesley Professional, 1st edition, [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-0-13-258220-9](https://en.wikipedia.org/wiki/Special:BookSources/978-0-13-258220-9)

Kafai, Y., & Resnick, M. (1996). *Constructionism in Practice: Designing, thinking and learning in a digital world LE.* New Jersey: Lawrence Erlbaum Associates.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Comput. Surv. 37, 2, 83-137.

Klarh, D. & Simon, H. A. (1999). Studies of Scientific discovery: Complementary approaches and convergent findings. *Psychological Bulletin, 125*, 524-543.

Knuth, Donald E. (1968), *The Art of Computer Programming*, Volume 1: Fundamental Algorithms. New Jersey: [Addison-Wesley](https://www.google.com/search?rlz=1C1CHBF_elGR831GR831&q=Addison-Wesley&stick=H4sIAAAAAAAAAOPgE-LQz9U3MC80qlDiArGMkyqMDfO01DLKrfST83NyUpNLMvPz9MuLMktKUvPiy_OLsoutCkqTcjKLM1KLFrHyOaakZBbn5-mGpxbnpFYCAA88BHxRAAAA&sa=X&ved=2ahUKEwjanrjuz6riAhUxMuwKHX4WBysQmxMoATAYegQIDhAO).

Kuhlman, D. (2009). *A Python Book: Beginning Python, Advanced Python, and Python Exercises*.  Open Source MIT License  (http://www.opensource.org/licenses/mit­license.php)

McIver, L., & Conway, D. (1996). Seven Deadly Sins of Introductory Programming Language Design. In Proceedings of the 1996 International Conference on Software Engineering: Education and Practice (SE:EP '96). IEEE Computer Society.

Minsky, M. (1967). *Computation: Finite and Infinite Machines (Automatic Computation)*. Prentice Hall

[*Oxford English Dictionary*](https://en.wikipedia.org/wiki/Oxford_English_Dictionary), Third Edition, 2012 [*s.v.*](http://www.oed.com/view/Entry/4959)

Papert, S. (1980). *Mindstorms. Children, computers and powerful ideas*, New York: Basic Books.

Papert, S. (1991). *Νοητικές θύελλες-Παιδιά, ηλεκτρονικοί υπολογιστές και δυναμικές ιδέες*, Αθήνα: Εκδόσεις Οδυσσέας

Piaget, J. (1954). *The construction of reality in the child*. New York: Basic Books.

Resnick, M. (1994). Changing the Centralized Mind. *Technology Review*, pp. 32-40. Ανακτήθηκε από: <http://web.media.mit.edu/~mres/papers/CentralizedMind/CentralizedMind.html>

Reusser, K. (1993). Tutoring systems and pedagogical theory: Representational tools for understanding, planning and reflection in problem solving. In S.P. Lajoie and S. J. Derry (eds.) Computers as cognitive tools, 143-177. London: LEA

Selby, C. C., & Woollard, J. (2013). *Computational Thinking: The Developing Definition*. University of Southampton.

Shein, E. (2015). Python for beginners. Commun. ACM 58, 3, 19-21.

Shunk, H. D. (2010). *Θεωρίες Μάθησης. Μια εκπαιδευτική προσέγγιση* (μτφ. Ε. Εκκεκάκη). Αθήνα: Μεταίχμιο.

Singh, J. K. and Zwirner, W. (1996). *Toward a theoretical framework of problem solving within Logo programming environments*. Journal of Research on Computing in Education, 29(1), 68-95

Tollervey, N. (2015). *Python in Education: Teach, Learn, Program*. USA: O’ Reilly Media

Vygotsky, L. S. (1978). *Interaction between Learning and Development*. In M. Cole, V. John-Steiner, S. Scribner, & E. Souberman (Eds.), Mind and Society: The Development of Higher Psychological Processes (pp. 79-91). Cambridge, MA: Harvard University Press.

Werner, L., Hanks, B., & McDowell, C. (2004). Pair programming helps female computer science students. *ACM Journal of Educational Resources in Computing, 4*(1), 1-8.

Wing, J. (2006). *Computational Thinking*, CACM, 49(3), 33-36

Wing, J. (2007). *Computational Thinking*

Wing, J. (2008). Computational thinking and thinking about computing, *Phil. Trans. R. Soc. A 366*, 3717-3725. doi:10.1098.

www.el.wikipedia.org (nd), *Αλγόριθμος*. Ανακτήθηκε από: <https://el.wikipedia.org/wiki/Αλγόριθμος>.

www.el.wikipedia.org (nd), *Διάγραμμα ροής*. Ανακτήθηκε από: <https://el.wikipedia.org/wiki/Διάγραμμα_ροής>.

www.el.wikipedia.org (nd), *Προγραμματισμός*. Ανακτήθηκε από: <https://el.wikipedia.org/wiki/Προγραμματισμός>.

[www.en.wikipedia.org](http://www.en.wikipedia.org) (nd), *Algorithm*. Ανακτήθηκε από: <https://en.wikipedia.org/wiki/Algorithm>.

www.en.wikipedia.org (nd), *Computer Programming*. Ανακτήθηκε από: <https://en.wikipedia.org/wiki/Computer_programming>.

www.en.wikipedia.org (nd), *Flowchart*. Ανακτήθηκε από: <https://en.wikipedia.org/wiki/Flowchart>

www.en.wikipedia.org (nd), *History of programming language*. Ανακτήθηκε από: <https://en.wikipedia.org/wiki/History_of_programming_languages>.

www.en.wikipedia.org (nd), *Programming language*. Ανακτήθηκε από: <https://en.wikipedia.org/wiki/Programming_language>.

[www.python.org](http://www.python.org) (nd), Python Software Foundation. Ανακτήθηκε από: <https://www.python.org/psf/>

www.tutorialspoint.com (nd), *Data Structures - Algorithms Basics*. Ανακτήθηκε από: <https://www.tutorialspoint.com/data_structures_algorithms/algorithms_basics.htm>.

Yelland, N. (1999). *Reconceptualising schooling with technology for the 21st century*. Information Technology in Childhood Education Annual, 1999(1), 39 (59).

Zhang, Y. & Luo, C. (2012). *Training for Computational Thinking Capability on Programming Language Teaching*. The 7th International Conference on Computer Science & Education (ICCSE 2012), 1804-1809. IEEE. doi: 10.1109/ICCSE.2012.6295420

Αράπογλου, Α., Βραχνός, Α., Κανίδης, Ε., Λέκκα, Δ., Μακρυγιάννης, Π., Μπελεσιώτης, Β., Παπαδάκης, Σπ., και Τζήμας, Δ. (2016). Προγραμματισμός Υπολογιστών – Γ’ ΕΠΑΛ. ΙΤΥΕ Διόφαντος.

Αράπογλου, Α., Βραχνός, Α., Κανίδης, Ε., Μακρυγιάννης, Π., Μπελεσιώτης, Β., και Τζήμας, Δ. (2015). Αρχές Προγραμματισμού Υπολογιστών – Β’ ΕΠΑΛ. ΙΤΥΕ Διόφαντος.

Αράπογλου, Α., Μαβόγλου, Χ., Οικονομάκος, Η., Φύτρος, Κ. (2006). Πληροφορική – Γ’ Γυμνασίου. Αθήνα: ΟΕΔΒ

Βραχνός, Ε. & Κατσένη Μ. (2017). *Αποτίμηση της πιλοτικής διδασκαλίας του προγραμματισμού με τη γλώσσα Python σε μαθητές Γυμνασίου*. In proceedings of the 9th Conference of Informatics in Education, Η πληροφορική στην εκπαίδευση. Επιμ. Αλεξανδρής, Ν., Δουληγέρης, Χ., Βλάμος, Π., Μπελεσιώτης, Β. ISBN: 978-960-578- 032-6

Ένωση Πληροφορικών Ελλάδας (2006). *Μελέτη επισκόπησης της Πληροφορικής* Ανακτήθηκε από: <https://www.epe.org.gr/meleth/final/MEP2006-3.pdf>

Κόμης, Β. (2004). *Εισαγωγή στις Εκπαιδευτικές Εφαρμογές των Τεχνολογιών της Πληροφορίας και των Επικοινωνιών*. Αθήνα: Νέων Τεχνολογιών.

Μικρόπουλος, Τ. Α. (2006). *Ο Υπολογιστής ως γνωστικό εργαλείο*, Αθήνα: ελληνικά γράμματα.

Παναγιωτακόπουλος, Χ. (2013). *Εισαγωγή στην επιστήμη των Ηλεκτρονικών Υπολογιστών*, Αθήνα: ίων.

Παναγιωτακόπουλος, Χ. (2013). *Οι Τεχνολογίες της Πληροφορίας και των Επικοινωνιών*, Αθήνα: ίων.

Ράπτης, Α., & Ράπτη, Α. (2007). *Μάθηση και διδασκαλία στην εποχή της Πληροφορίας ολική προσέγγιση.* Αθήνα: Έκδοση Ιδίων.

ΥΠΕΠΘ (1997). *Ενιαίο Πλαίσιο Προγράμματος Σπουδών Πληροφορικής*, Παιδαγωγικό Ινστιτούτο.

ΥΠΕΠΘ (2003). *Διαθεματικό Ενιαίο Πλαίσιο Προγράμματος Σπουδών Πληροφορικής*, Παιδαγωγικό Ινστιτούτο.

Ψυχάρης, Σ. & Καλοβρέκτης, Κ. (2018). *Διδακτική & Σχεδιασμός Εκπαιδευτικών δραστηριοτήτων STEM & ΤΠΕ*. Αθήνα: Εκδόσεις Τζιόλα