Βασικότατες εντολές της Python

Απόστολος Συρόπουλος

Ξάνθη
asyropoulos@Gargoyle.com

Βασικότατες εντολές της Python
για μαθητές Γυμνασίου

Περίγραμμα παρουσίασης

Εισαγωγή

Τιμές και μεταβλητές

Εντολές ανάθεσης και εισόδου/εξόδου

Εντολές ελέγχου

Ανακεφαλαίωση

Τι είναι η Python;

Τι είναι η Python;

- Είναι μια πολύ δημοφιλής γλώσσα προγραμματισμού.

Τι είναι η Python;

- Είναι μια πολύ δημοφιλής γλώσσα προγραμματισμού.
- Δημιουργήθηκε από τον Ολλανδό προγραμματιστή Guido van Rossum το 1991.

Τι είναι η Python;

- Είναι μια πολύ δημοφιλής γλώσσα προγραμματισμού.
- Δημιουργήθηκε από τον Ολλανδό προγραμματιστή Guido van Rossum το 1991.
- Η γλώσσα επιρρεάστηκε από ABC, μια παλαιότερη γλώσσα προγραμματισμού.

Τι είναι η Python;

- Είναι μια πολύ δημοφιλής γλώσσα προγραμματισμού.
- Δημιουργήθηκε από τον Ολλανδό προγραμματιστή Guido van Rossum το 1991.
- Η γλώσσα επιρρεάστηκε από ABC, μια παλαιότερη γλώσσα προγραμματισμού.
- Εδώ και χρόνια χρησιμοποιείται για τη διδασκαλία του προγραμματισμού σε πολλές χώρες.

Τι είναι η Python;

- Είναι μια πολύ δημοφιλής γλώσσα προγραμματισμού.
- Δημιουργήθηκε από τον Ολλανδό προγραμματιστή Guido van Rossum το 1991.
- Η γλώσσα επιρρεάστηκε από ABC, μια παλαιότερη γλώσσα προγραμματισμού.
- Εδώ και χρόνια χρησιμοποιείται για τη διδασκαλία του προγραμματισμού σε πολλές χώρες.
- Παρέχει μεγάλη βιβλιοθήκη (επιπρόσθετες δυνατότητες) που την καθιστούν ιδιαίτερα εκλυστική.

Τι είναι η Python;

- Είναι μια πολύ δημοφιλής γλώσσα προγραμματισμού.
- Δημιουργήθηκε από τον Ολλανδό προγραμματιστή Guido van Rossum το 1991.
- Η γλώσσα επιρρεάστηκε από ABC, μια παλαιότερη γλώσσα προγραμματισμού.
- Εδώ και χρόνια χρησιμοποιείται για τη διδασκαλία του προγραμματισμού σε πολλές χώρες.
- Παρέχει μεγάλη βιβλιοθήκη (επιπρόσθετες δυνατότητες) που την καθιστούν ιδιαίτερα εκλυστική.
- Υποστηρίζει ταυτόχρονα διάφορες προγραμματιστικές φιλοσοφίες.

Τιμές

- Μια τιμή είναι…

Τιμές

- Μια τιμή είναι…
- Ένας ακέραιος ή δεκαδικός αριθμός, δηλαδή μια ποσότητα όπως το `41` ή `3.14159`, αντίστοιχα. Ο δεκαδικός αριθμός 4 γραφεται `4.0`.

Τιμές

- Μια τιμή είναι…
- Ένας ακέραιος ή δεκαδικός αριθμός, δηλαδή μια ποσότητα όπως το `41` ή `3.14159`, αντίστοιχα. Ο δεκαδικός αριθμός 4 γραφεται `4.0`.
- Μια **λογική τιμή**, δηλαδή οι τιμές `True` (**αληθές**) και `False` (**ψευδές**). Αυτές οι τιμές είναι απαντήσεις σε ερωτήσεις της μορφής $3 \neq 4$ ή $3 < 4$.

Τιμές

- Μια τιμή είναι…
- Ένας ακέραιος ή δεκαδικός αριθμός, δηλαδή μια ποσότητα όπως το `41` ή `3.14159`, αντίστοιχα. Ο δεκαδικός αριθμός 4 γραφεται `4.0`.
- Μια **λογική τιμή**, δηλαδή οι τιμές `True` (**αληθές**) και `False` (**ψευδές**). Αυτές οι τιμές είναι απαντήσεις σε ερωτήσεις της μορφής $3 \neq 4$ ή $3 < 4$.
- Μια ακολουθία **χαρακτήρων** που περικλείεται από τα σύμβολα `"` και `"` ή τα σύμβολα `'` και `'` ονομάζεται αλφαριθμητική τιμή ή απλά στρίνγκ (**string**). Παραδείγματα: `"Help!"` και `'Life'`.

Μεταβλητές

Μεταβλητές

- Μια μεταβλητή είναι μια λέξη που χρησιμοποιείται αντί μιας τιμής.

Μεταβλητές

- Μια μεταβλητή είναι μια λέξη που χρησιμοποιείται αντί μιας τιμής.
- Στα Μαθηματικά γράφουμε π για να δηλώσουμε την τιμή $3.1415926535897\ldots$.

Μεταβλητές

- Μια μεταβλητή είναι μια λέξη που χρησιμοποιείται αντί μιας τιμής.
- Στα Μαθηματικά γράφουμε π για να δηλώσουμε την τιμή $3.1415926535897\ldots$.
- Μια μεταβλητή είναι το όνομα μιας θέσης μνήμης στην οποία μπορούμε να αποθηκεύσουμε τιμές.

Μεταβλητές

- Μια μεταβλητή είναι μια λέξη που χρησιμοποιείται αντί μιας τιμής.
- Στα Μαθηματικά γράφουμε π για να δηλώσουμε την τιμή $3.1415926535897\ldots$.
- Μια μεταβλητή είναι το όνομα μιας θέσης μνήμης στην οποία μπορούμε να αποθηκεύσουμε τιμές.
- Καθώς περνάει ο χρόνος, μια μεταβλητή μπορεί να «περιέχει» διαφορετικές τιμές.

Μεταβλητές

- Μια μεταβλητή είναι μια λέξη που χρησιμοποιείται αντί μιας τιμής.
- Στα Μαθηματικά γράφουμε π για να δηλώσουμε την τιμή $3.1415926535897\ldots$.
- Μια μεταβλητή είναι το όνομα μιας θέσης μνήμης στην οποία μπορούμε να αποθηκεύσουμε τιμές.
- Καθώς περνάει ο χρόνος, μια μεταβλητή μπορεί να «περιέχει» διαφορετικές τιμές.
- Παράδειγμα: Μια διεργασία που μετράει αντικείμενα. Μια μεταβλητή που «κρατάει» τον αριθμό αντικειμένων που μετρήθηκαν, θα έχει διαφορετικές τιμες σε διαφορετικές στιγμές.

Μεταβλητές

- Μια μεταβλητή είναι μια λέξη που χρησιμοποιείται αντί μιας τιμής.
- Στα Μαθηματικά γράφουμε π για να δηλώσουμε την τιμή 3.1415926535897 ....
- Μια μεταβλητή είναι το όνομα μιας θέσης μνήμης στην οποία μπορούμε να αποθηκεύσουμε τιμές.
- Καθώς περνάει ο χρόνος, μια μεταβλητή μπορεί να «περιέχει» διαφορετικές τιμές.
- Παράδειγμα: Μια διεργασία που μετράει αντικείμενα. Μια μεταβλητή που «κρατάει» τον αριθμό αντικειμένων που μετρήθηκαν, θα έχει διαφορετικές τιμες σε διαφορετικές στιγμές.
- Ο πρώτος χαρακτήρας του ονόματος πρέπει να είναι Αγγλικό γράμμα. Προσοχή! Οι μεταβλητές `A` και `a` θεωρούνται διαφορετικές.

Μεταβλητές

- Μια μεταβλητή είναι μια λέξη που χρησιμοποιείται αντί μιας τιμής.
- Στα Μαθηματικά γράφουμε π για να δηλώσουμε την τιμή 3.1415926535897 ....
- Μια μεταβλητή είναι το όνομα μιας θέσης μνήμης στην οποία μπορούμε να αποθηκεύσουμε τιμές.
- Καθώς περνάει ο χρόνος, μια μεταβλητή μπορεί να «περιέχει» διαφορετικές τιμές.
- Παράδειγμα: Μια διεργασία που μετράει αντικείμενα. Μια μεταβλητή που «κρατάει» τον αριθμό αντικειμένων που μετρήθηκαν, θα έχει διαφορετικές τιμες σε διαφορετικές στιγμές.
- Ο πρώτος χαρακτήρας του ονόματος πρέπει να είναι Αγγλικό γράμμα. Προσοχή! Οι μεταβλητές `A` και `a` θεωρούνται διαφορετικές.
- Αν υπάρχουν και άλλοι χαρακτήρες, μπορούν να είναι γράμματα ή ψηφία.

Χρησιμοποιώντας μεταβλητές

Χρησιμοποιώντας μεταβλητές

- Μια μεταβλητη εισαγάγεται με μία εντολή ανάθεσης.

Χρησιμοποιώντας μεταβλητές

- Μια μεταβλητη εισαγάγεται με μία εντολή ανάθεσης.
- Ανάθεση: *μεταβλητή = τιμή*.

Χρησιμοποιώντας μεταβλητές

- Μια μεταβλητη εισαγάγεται με μία εντολή ανάθεσης.
- Ανάθεση: *μεταβλητή = τιμή*.
- Η εντολή `x = 3` αναθέτει ή δίνει την τιμή `3` στη μεταβλητή `x`.

Χρησιμοποιώντας μεταβλητές

- Μια μεταβλητη εισαγάγεται με μία εντολή ανάθεσης.
- Ανάθεση: *μεταβλητή = τιμή*.
- Η εντολή `x = 3` αναθέτει ή δίνει την τιμή `3` στη μεταβλητή `x`.
- Η εντολή `x = x + 1` αναθέτει ή δίνει στη μεταβλητή `x` την τιμή που έχει τώρα η μεταβλητή `x` αφού πρώτα προσθέσει σε αυτή την τιμή `1`.

Χρησιμοποιώντας μεταβλητές

- Μια μεταβλητη εισαγάγεται με μία εντολή ανάθεσης.
- Ανάθεση: *μεταβλητή = τιμή*.
- Η εντολή `x = 3` αναθέτει ή δίνει την τιμή `3` στη μεταβλητή `x`.
- Η εντολή `x = x + 1` αναθέτει ή δίνει στη μεταβλητή `x` την τιμή που έχει τώρα η μεταβλητή `x` αφού πρώτα προσθέσει σε αυτή την τιμή `1`.
- Βρείτε τις τιμές των μεταβλητών μετά την εκτέλεση των παρακάτω εντολών

```
x=4
y=6
z=2*x # με * σημειώνουμε το επί
x=x+1+y # με + σημειώνουμε το συν
```

Αριθμητικές πράξεις

Αριθμητικές πράξεις

- Η πρόσθεση δύο τιμών σημειώνεται με $τιμή_1$ + $τιμή_2$, π.χ. `4 + y`.

Αριθμητικές πράξεις

- Η πρόσθεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1 + \tau\iota\mu\acute{\eta}_2$, π.χ. `4 + y`.
- Η αφαίρεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1 - \tau\iota\mu\acute{\eta}_2$, π.χ. `3 - 2`.

Αριθμητικές πράξεις

- Η πρόσθεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1 + \tau\iota\mu\acute{\eta}_2$, π.χ. `4 + y`.
- Η αφαίρεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1 - \tau\iota\mu\acute{\eta}_2$, π.χ. `3 - 2`.
- Ο πολλαπλασιασμός δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1 * \tau\iota\mu\acute{\eta}_2$, π.χ. `5*6`.

Αριθμητικές πράξεις

- Η πρόσθεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ `+` $\tau\iota\mu\acute{\eta}_2$, π.χ. `4 + y`.
- Η αφαίρεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ `-` $\tau\iota\mu\acute{\eta}_2$, π.χ. `3 - 2`.
- Ο πολλαπλασιασμός δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ `*` $\tau\iota\mu\acute{\eta}_2$, π.χ. `5*6`.
- Το δεκαδικό πηλίκο δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ `/` $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `3 / 2` ισούται με `1.5`.

Αριθμητικές πράξεις

- Η πρόσθεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ `+` $\tau\iota\mu\acute{\eta}_2$, π.χ. `4 + y`.
- Η αφαίρεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ `-` $\tau\iota\mu\acute{\eta}_2$, π.χ. `3 - 2`.
- Ο πολλαπλασιασμός δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ `*` $\tau\iota\mu\acute{\eta}_2$, π.χ. `5*6`.
- Το δεκαδικό πηλίκο δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ `/` $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `3 / 2` ισούται με `1.5`.
- Το ακέραιο πηλίκο δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ `//` $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `3 // 2` ισούται με `1`.

Αριθμητικές πράξεις

- Η πρόσθεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ + $\tau\iota\mu\acute{\eta}_2$, π.χ. `4 + y`.
- Η αφαίρεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ − $\tau\iota\mu\acute{\eta}_2$, π.χ. `3 - 2`.
- Ο πολλαπλασιασμός δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ * $\tau\iota\mu\acute{\eta}_2$, π.χ. `5*6`.
- Το δεκαδικό πηλίκο δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ / $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `3 / 2` ισούται με `1.5`.
- Το ακέραιο πηλίκο δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ // $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `3 // 2` ισούται με `1`.
- Το υπόλοιπο της διαίρεσης δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ % $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `5 % 2` ισούται με `1`.

Αριθμητικές πράξεις

- Η πρόσθεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ + $\tau\iota\mu\acute{\eta}_2$, π.χ. `4 + y`.
- Η αφαίρεση δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ – $\tau\iota\mu\acute{\eta}_2$, π.χ. `3 - 2`.
- Ο πολλαπλασιασμός δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ * $\tau\iota\mu\acute{\eta}_2$, π.χ. `5*6`.
- Το δεκαδικό πηλίκο δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ / $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `3 / 2` ισούται με `1.5`.
- Το ακέραιο πηλίκο δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ // $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `3 // 2` ισούται με `1`.
- Το υπόλοιπο της διαίρεσης δύο τιμών σημειώνεται με $\tau\iota\mu\acute{\eta}_1$ % $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `5 % 2` ισούται με `1`.
- Με την παράσταση $\tau\iota\mu\acute{\eta}_1$ ** $\tau\iota\mu\acute{\eta}_2$ σημειωνουμε την ύψωση της τιμής $\tau\iota\mu\acute{\eta}_1$ στη δύναμη $\tau\iota\mu\acute{\eta}_2$, π.χ. η παράσταση `3 ** 3` ισούται με `27`.

Πράξεις μεταξύ λογικών τιμών

Πράξεις μεταξύ λογικών τιμών

- Η παράσταση $\tau\iota\mu\acute{\eta}_1$ `and` $\tau\iota\mu\acute{\eta}_2$ είναι `True` μόνο όταν η $\tau\iota\mu\acute{\eta}_1$ και η $\tau\iota\mu\acute{\eta}_2$ είναι `True`.

Πράξεις μεταξύ λογικών τιμών

- Η παράσταση $\tau\iota\mu\acute{\eta}_1$ `and` $\tau\iota\mu\acute{\eta}_2$ είναι `True` μόνο όταν η $\tau\iota\mu\acute{\eta}_1$ και η $\tau\iota\mu\acute{\eta}_2$ είναι `True`.
- Η παράσταση $\tau\iota\mu\acute{\eta}_1$ `or` $\tau\iota\mu\acute{\eta}_2$ είναι `True` μόνο όταν είτε η $\tau\iota\mu\acute{\eta}_1$ είτε η $\tau\iota\mu\acute{\eta}_2$ είναι `True`.

Πράξεις μεταξύ λογικών τιμών

- Η παράσταση $\tau\iota\mu\acute{\eta}_1$ `and` $\tau\iota\mu\acute{\eta}_2$ είναι `True` μόνο όταν η $\tau\iota\mu\acute{\eta}_1$ και η $\tau\iota\mu\acute{\eta}_2$ είναι `True`.
- Η παράσταση $\tau\iota\mu\acute{\eta}_1$ `or` $\tau\iota\mu\acute{\eta}_2$ είναι `True` μόνο όταν είτε η $\tau\iota\mu\acute{\eta}_1$ είτε η $\tau\iota\mu\acute{\eta}_2$ είναι `True`.
- Η παράσταση `not` $\tau\iota\mu\acute{\eta}$ είναι `True` μόνο όταν η $\tau\iota\mu\acute{\eta}$ είναι `False` και `False` μόνο όταν η $\tau\iota\mu\acute{\eta}$ είναι `True`.

Πράξεις μεταξύ λογικών τιμών

- Η παράσταση $\tau\iota\mu\acute{\eta}_1$ `and` $\tau\iota\mu\acute{\eta}_2$ είναι `True` μόνο όταν η $\tau\iota\mu\acute{\eta}_1$ και η $\tau\iota\mu\acute{\eta}_2$ είναι `True`.
- Η παράσταση $\tau\iota\mu\acute{\eta}_1$ `or` $\tau\iota\mu\acute{\eta}_2$ είναι `True` μόνο όταν είτε η $\tau\iota\mu\acute{\eta}_1$ είτε η $\tau\iota\mu\acute{\eta}_2$ είναι `True`.
- Η παράσταση `not` $\tau\iota\mu\acute{\eta}$ είναι `True` μόνο όταν η $\tau\iota\mu\acute{\eta}$ είναι `False` και `False` μόνο όταν η $\tau\iota\mu\acute{\eta}$ είναι `True`.
- Παραδείγματα.

```
>>> True and False
False
>>> True and True
True
>>> False or True
True
>>> False or False
False
>>> not False
True
```

Συγκρίσεις

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),
  - `>` (μεγαλύτερο από),

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),
  - `>` (μεγαλύτερο από),
  - `<=` (μικρότερο ή ίσο από) και

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),
  - `>` (μεγαλύτερο από),
  - `<=` (μικρότερο ή ίσο από) και
  - `>=` (μεγαλύτερο ή ίσο προς).

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),
  - `>` (μεγαλύτερο από),
  - `<=` (μικρότερο ή ίσο από) και
  - `>=` (μεγαλύτερο ή ίσο προς).
- Υποθέστε πως οι εντολές `x = 5` και `y = 7` μόλις εκτελέστηκαν, τότε:

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),
  - `>` (μεγαλύτερο από),
  - `<=` (μικρότερο ή ίσο από) και
  - `>=` (μεγαλύτερο ή ίσο προς).
- Υποθέστε πως οι εντολές `x = 5` και `y = 7` μόλις εκτελέστηκαν, τότε:
  - η παράσταση `x == 5` έχει την τιμή `True`.

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),
  - `>` (μεγαλύτερο από),
  - `<=` (μικρότερο ή ίσο από) και
  - `>=` (μεγαλύτερο ή ίσο προς).
- Υποθέστε πως οι εντολές `x = 5` και `y = 7` μόλις εκτελέστηκαν, τότε:
  - η παράσταση `x == 5` έχει την τιμή `True`·
  - η παράσταση `x != 5` έχει την τιμή `False`·

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),
  - `>` (μεγαλύτερο από),
  - `<=` (μικρότερο ή ίσο από) και
  - `>=` (μεγαλύτερο ή ίσο προς).
- Υποθέστε πως οι εντολές `x = 5` και `y = 7` μόλις εκτελέστηκαν, τότε:
  - η παράσταση `x == 5` έχει την τιμή `True`·
  - η παράσταση `x != 5` έχει την τιμή `False`·
  - η παράσταση `x < y` έχει την τιμή `True`·

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),
  - `>` (μεγαλύτερο από),
  - `<=` (μικρότερο ή ίσο από) και
  - `>=` (μεγαλύτερο ή ίσο προς).
- Υποθέστε πως οι εντολές `x = 5` και `y = 7` μόλις εκτελέστηκαν, τότε:
  - η παράσταση `x == 5` έχει την τιμή `True`·
  - η παράσταση `x != 5` έχει την τιμή `False`·
  - η παράσταση `x < y` έχει την τιμή `True`·
  - η παράσταση `y >= 7` έχει την τιμή `True`·

Συγκρίσεις

- Οι τελεστές σύγκρισης είναι σύμβολα τα οποία εκτελούν πράξεις σε δεδομένα και έχουν ως αποτέλσμα `True` ή `False` ανάλογα με τα δεδομένα και το είδος της σύγκρισης.
- Οι παρακάτω είναι βασικοί τελεστές σύγκρισης
  - `==` (ίσο προς),
  - `!=` (άνισο ή μη ίσο προς),
  - `<` (μικρότερο από),
  - `>` (μεγαλύτερο από),
  - `<=` (μικρότερο ή ίσο από) και
  - `>=` (μεγαλύτερο ή ίσο προς).
- Υποθέστε πως οι εντολές `x = 5` και `y = 7` μόλις εκτελέστηκαν, τότε:
  - η παράσταση `x == 5` έχει την τιμή `True`·
  - η παράσταση `x != 5` έχει την τιμή `False`·
  - η παράσταση `x < y` έχει την τιμή `True`·
  - η παράσταση `y >= 7` έχει την τιμή `True`·
  - η παράσταση `x <= 5` έχει την τιμή `True`.

Εντολή εξόδου

Εντολή εξόδου

- Με μία εντολη εξόδου τυπώνουμε την τιμή μιας μεταβλητής ή το αποτέλεσμα μιας πράξης, δηλαδή μία τιμή.

Εντολή εξόδου

- Με μία εντολη εξόδου τυπώνουμε την τιμή μιας μεταβλητής ή το αποτέλεσμα μιας πράξης, δηλαδή μία τιμή.
- Εντολή εξόδου: `print(μεταβλητή ή τιμή)`.

Εντολή εξόδου

- Με μία εντολη εξόδου τυπώνουμε την τιμή μιας μεταβλητής ή το αποτέλεσμα μιας πράξης, δηλαδή μία τιμή.
- Εντολή εξόδου: `print(`*`μεταβλητή ή τιμή`*`)`.
- Οι εντολές

```
x = 4
print(x)
```

θα έχουν ως αποτέλεσμα να τυπώθεί ο αριθμός 4 στην οθόνη του υπολογιστή.

Εντολή εξόδου

- Με μία εντολη εξόδου τυπώνουμε την τιμή μιας μεταβλητής ή το αποτέλεσμα μιας πράξης, δηλαδή μία τιμή.
- Εντολή εξόδου: `print(`*`μεταβλητή ή τιμή`*`)`.
- Οι εντολές

```
x = 4
print(x)
```

  θα έχουν ως αποτέλεσμα να τυπώθεί ο αριθμός 4 στην οθόνη του υπολογιστή.
- Μπορούμε να τυπώσουμε πολλές τιμές μεταβλητών ή απλές τιμές βάζοντας μεταξύ τους ένα κόμμα:

```
x = 4
y = 5
print(x,", ",y)
```

  Η τελευταία εντολή θα τυπώσει `4 , 5`

Εντολές εισόδου

Εντολές εισόδου

- Με μία εντολη εισόδου το πρόγραμμά μας «διαβάζει» μια τιμή από το πληκτρολόγιο.

Εντολές εισόδου

- Με μία εντολη εισόδου το πρόγραμμά μας «διαβάζει» μια τιμή από το πληκτρολόγιο.
- Εντολή εισόδου: `μεταβλητή = input(προτροπή)`, η *προτροπή* είναι ένα στρινγκ που εξηγεί τι περίμένει το πρόγραμμα να πληκτρολογήσει ο χρήστης. Στη *μεταβλητή* ανατίθεται ως τιμή τύπου στρινγκ ότι πληκτρολογήσει ο χρήστης.

Εντολές εισόδου

- Με μία εντολη εισόδου το πρόγραμμά μας «διαβάζει» μια τιμή από το πληκτρολόγιο.
- Εντολή εισόδου: `μεταβλητή = input(προτροπή)`, η *προτροπή* είναι ένα στρινγκ που εξηγεί τι περίμένει το πρόγραμμα να πληκτρολογήσει ο χρήστης. Στη *μεταβλητή* ανατίθεται ως τιμή τύπου στρινγκ ότι πληκτρολογήσει ο χρήστης.
- Παράδειγμα:

```
x = input("Δώσε έναν αριθμό... ")
```

Εντολές εισόδου

- Με μία εντολη εισόδου το πρόγραμμά μας «διαβάζει» μια τιμή από το πληκτρολόγιο.
- Εντολή εισόδου: `μεταβλητή = input(προτροπή)`, η *προτροπή* είναι ένα στρινγκ που εξηγεί τι περίμένει το πρόγραμμα να πληκτρολογήσει ο χρήστης. Στη *μεταβλητή* ανατίθεται ως τιμή τύπου στρινγκ ότι πληκτρολογήσει ο χρήστης.
- Παράδειγμα:

```
x = input("Δώσε έναν αριθμό... ")
```

- Μπορούμε να «υποχρεώσσουμε» την Python να «διαβάσει» αριθμούς αν χρησιμοποιήσουμε το ιδίωμα `μεταβλητή = eval(input(προτροπή))`. Παράδειγμα:

Εντολές εισόδου

- Με μία εντολη εισόδου το πρόγραμμά μας «διαβάζει» μια τιμή από το πληκτρολόγιο.
- Εντολή εισόδου: `μεταβλητή = input(προτροπή)`, η *προτροπή* είναι ένα στρινγκ που εξηγεί τι περίμένει το πρόγραμμα να πληκτρολογήσει ο χρήστης. Στη *μεταβλητή* ανατίθεται ως τιμή τύπου στρινγκ ότι πληκτρολογήσει ο χρήστης.
- Παράδειγμα:

```
x = input("Δώσε έναν αριθμό... ")
```

- Μπορούμε να «υποχρεώσσουμε» την Python να «διαβάσει» αριθμούς αν χρησιμοποιήσουμε το ιδίωμα `μεταβλητή = eval(input(προτροπή))`. Παράδειγμα:

```
>>> x = eval(input("Δώσε έναν αριθμό... "))
Δώσε έναν αριθμό... 7+9
>>> x+10
26
>>>
```

Η εντολή `if`

Η εντολή `if` έχει δύο μορφές:

Η εντολή if

Η εντολή `if` έχει δύο μορφές:

```
if λογική τιμή:
  εντολή_1
  εντολή_2
    ⋮
  εντολή_n
```

Τύπος (α)

```
if λογική τιμή:
  εντολή_1
    ⋮
else:
  εντολή'_1
    ⋮
```

Τύπος (β)

Η εντολή `if`

Η εντολή `if` έχει δύο μορφές:

```
if λογική τιμή:
  εντολή_1
  εντολή_2
    ⋮
  εντολή_n
```

Τύπος (α)

```
if λογική τιμή:
  εντολή_1
    ⋮
else:
  εντολή'_1
    ⋮
```

Τύπος (β)

- Στον τύπο (α) γίνεται έλεγχος της *λογικής τιμής*:

Η εντολή `if`

Η εντολή `if` έχει δύο μορφές:

```
if λογική τιμή:
    εντολή_1
    εντολή_2
      ⋮
    εντολή_n
```

Τύπος (α)

```
if λογική τιμή:
    εντολή_1
      ⋮
else:
    εντολή'_1
      ⋮
```

Τύπος (β)

- Στον τύπο (α) γίνεται έλεγχος της *λογικής τιμής*:
  - Αν είναι `True`, εκτελουνται οι *εντολές*.

Η εντολή `if`

Η εντολή `if` έχει δύο μορφές:

```
if λογική τιμή:
  εντολή_1
  εντολή_2
    ⋮
  εντολή_n
```

Τύπος (α)

```
if λογική τιμή:
  εντολή_1
    ⋮
else:
  εντολή'_1
    ⋮
```

Τύπος (β)

- Στον τύπο (α) γίνεται έλεγχος της *λογικής τιμής*:
  - Αν είναι `True`, εκτελούνται οι *εντολές*.
  - Αλλιώς δεν συμβαίνει τίποτα.

Η εντολή `if`

Η εντολή `if` έχει δύο μορφές:

```
if λογική τιμή:
    εντολή_1
    εντολή_2
      ⋮
    εντολή_n
```

Τύπος (α)

```
if λογική τιμή:
    εντολή_1
      ⋮
else:
    εντολή'_1
      ⋮
```

Τύπος (β)

- Στον τύπο (α) γίνεται έλεγχος της *λογικής τιμής*:
  - Αν είναι `True`, εκτελουνται οι *εντολές*·
  - Αλλιώς δεν συμβαίνει τίποτα.
- Στον τύπο (β) αν η *λογική τιμή* είναι `False`, εκτελούνται οι *εντολές'*.

Παράδειγμα της εντολής `if`

Παράδειγμα της εντολής `if`

- Ας γράψουμε ένα απλό πρόγραμμα που θα διαβάζει έναν ακέραιο αριθμό και θα ελέγχει αν είναι άρτιος ή περιττός.

Παράδειγμα της εντολής `if`

- Ας γράψουμε ένα απλό πρόγραμμα που θα διαβάζει έναν ακέραιο αριθμό και θα ελέγχει αν είναι άρτιος ή περιττός.
- Υπενθύμιση: Ένας ακέραιος αριθμός είναι άρτιος αν το υπόλοιπο της διαίρεσης του αριθμού με το δύο είναι μηδέν.

Παράδειγμα της εντολής `if`

- Ας γράψουμε ένα απλό πρόγραμμα που θα διαβάζει έναν ακέραιο αριθμό και θα ελέγχει αν είναι άρτιος ή περιττός.
- Υπενθύμιση: Ένας ακέραιος αριθμός είναι άρτιος αν το υπόλοιπο της διαίρεσης του αριθμού με το δύο είναι μηδέν.
- Ακολουθεί ο κώδικας:

Παράδειγμα της εντολής `if`

- Ας γράψουμε ένα απλό πρόγραμμα που θα διαβάζει έναν ακέραιο αριθμό και θα ελέγχει αν είναι άρτιος ή περιττός.
- Υπενθύμιση: Ένας ακέραιος αριθμός είναι άρτιος αν το υπόλοιπο της διαίρεσης του αριθμού με το δύο είναι μηδέν.
- Ακολουθεί ο κώδικας:

```
x = eval(input("Δώσε έναν ακέραιο αριθμό...? "))
if x % 2 == 0:
  print("ο αριθμός", x,"είναι άρτιος")
else:
  print("ο αριθμός", x,"είναι περιττός")
```

Παράδειγμα της εντολής `if`

- Ας γράψουμε ένα απλό πρόγραμμα που θα διαβάζει έναν ακέραιο αριθμό και θα ελέγχει αν είναι άρτιος ή περιττός.
- Υπενθύμιση: Ένας ακέραιος αριθμός είναι άρτιος αν το υπόλοιπο της διαίρεσης του αριθμού με το δύο είναι μηδέν.
- Ακολουθεί ο κώδικας:

```
x = eval(input("Δώσε έναν ακέραιο αριθμό...? "))
if x % 2 == 0:
  print("ο αριθμός", x,"είναι άρτιος")
else:
  print("ο αριθμός", x,"είναι περιττός")
```

- Προσέξτε ότι κάποιες εντολές πληκτρολογούνται πιο «μέσα»!

Παράδειγμα της εντολής `if`

- Ας γράψουμε ένα απλό πρόγραμμα που θα διαβάζει έναν ακέραιο αριθμό και θα ελέγχει αν είναι άρτιος ή περιττός.
- Υπενθύμιση: Ένας ακέραιος αριθμός είναι άρτιος αν το υπόλοιπο της διαίρεσης του αριθμού με το δύο είναι μηδέν.
- Ακολουθεί ο κώδικας:

```
x = eval(input("Δώσε έναν ακέραιο αριθμό...? "))
if x % 2 == 0:
  print("ο αριθμός", x,"είναι άρτιος")
else:
  print("ο αριθμός", x,"είναι περιττός")
```

- Προσέξτε ότι κάποιες εντολές πληκτρολογούνται πιο «μέσα»!
- Εξηγήστε τι κάνει ο παραπάνω κώδικας!

Η εντολη `for`

Η εντολη `for`

- Η πιο απλά μορφή της εντολής είναι η ακόλουθη:

Η εντολη `for`

- Η πιο απλά μορφή της εντολής είναι η ακόλουθη:
- ```
  for μεταβλητή in range(αρχή,τέλος):
      εντολές
  ```
  ```

Η εντολη `for`

- Η πιο απλά μορφή της εντολής είναι η ακόλουθη:
- ```
  for μεταβλητή in range(αρχή,τέλος):
      εντολές
  ```
- Η *μεταβλητή* διατρέχει όλες τις τιμές από *αρχή* ως *τέλος*−1.
  ```

Η εντολη `for`

- Η πιο απλά μορφή της εντολής είναι η ακόλουθη:
- ```
  for μεταβλητή in range(αρχή,τέλος):
      εντολές
  ```
- Η *μεταβλητή* διατρέχει όλες τις τιμές από *αρχή* ως *τέλος*`-1`.
- Η πιο γενική μορφή της εντολής είναι η εξής:
  ```

Η εντολη `for`

- Η πιο απλά μορφή της εντολής είναι η ακόλουθη:
- ```
  for μεταβλητή in range(αρχή,τέλος):
      εντολές
  ```
- Η *μεταβλητή* διατρέχει όλες τις τιμές από *αρχή* ως *τέλος*`-1`.
- Η πιο γενική μορφή της εντολής είναι η εξής:
- ```
  for μεταβλητή in range(αρχή,τέλος,βήμα):
      εντολές
  ```

Η εντολη `for`

- Η πιο απλά μορφή της εντολής είναι η ακόλουθη:
- ```
  for μεταβλητή in range(αρχή,τέλος):
    εντολές
  ```
- Η *μεταβλητή* διατρέχει όλες τις τιμές από *αρχή* ως *τέλος*`-1`.
- Η πιο γενική μορφή της εντολής είναι η εξής:
- ```
  for μεταβλητή in range(αρχή,τέλος,βήμα):
    εντολές
  ```
- Η διαφορά εδώ είναι ότι η *μεταβλητή* διατρέχει τις τιμές ανά *βήμα*.

Παραδείγματα της εντολής `for`

Παραδείγματα της εντολής `for`

- Ας γράψουμε ένα απλό πρόγραμμα που θα τυπώνει τους αριθμούς από το 1 ως το 10.

Παραδείγματα της εντολής `for`

- Ας γράψουμε ένα απλό πρόγραμμα που θα τυπώνει τους αριθμούς από το 1 ως το 10.

```
for i in range(1,11):
  print(i)
```

Παραδείγματα της εντολής `for`

- Ας γράψουμε ένα απλό πρόγραμμα που θα τυπώνει τους αριθμούς από το 1 ως το 10.

```
for i in range(1,11):
  print(i)
```

- Ας γράψουμε τώρα ένα πρόγραμμα που θα τυπώνει τους περιττούς αριθμούς από το 1 ως το 10.

Παραδείγματα της εντολής `for`

- Ας γράψουμε ένα απλό πρόγραμμα που θα τυπώνει τους αριθμούς από το 1 ως το 10.

```
for i in range(1,11):
  print(i)
```

- Ας γράψουμε τώρα ένα πρόγραμμα που θα τυπώνει τους περιττούς αριθμούς από το 1 ως το 10.

```
for i in range(1,11,2):
  print(i)
```

Παραδείγματα της εντολής `for`

- Ας γράψουμε ένα απλό πρόγραμμα που θα τυπώνει τους αριθμούς από το 1 ως το 10.

```
for i in range(1,11):
  print(i)
```

- Ας γράψουμε τώρα ένα πρόγραμμα που θα τυπώνει τους περιττούς αριθμούς από το 1 ως το 10.

```
for i in range(1,11,2):
  print(i)
```

- Εξηγήστε τι κάνει ο παραπάνω κώδικας!

Η εντολη `while`

Η εντολη `while`

- Η εντολή `while` εκτελεί μια σειρά εντολών όσο μια *συνθήκη*, που έχει ως αποτέλεσμα μια λογική τιμή, είναι `True`.

Η εντολη `while`

- Η εντολή `while` εκτελεί μια σειρά εντολών όσο μια *συνθήκη*, που έχει ως αποτέλεσμα μια λογική τιμή, είναι `True`.
- ```
  while συνθήκη:
      εντολές
  ```
  ```

Η εντολη `while`

- Η εντολή `while` εκτελεί μια σειρά εντολών όσο μια *συνθήκη*, που έχει ως αποτέλεσμα μια λογική τιμή, είναι `True`.
- ```
  while συνθήκη:
      εντολές
  ```
- Αρχικά ελέγχεται η *συνθήκη* και αν είναι `True` εκτελούνται οι *εντολές*.
  ```

Η εντολη `while`

- Η εντολή `while` εκτελεί μια σειρά εντολών όσο μια *συνθήκη*, που έχει ως αποτέλεσμα μια λογική τιμή, είναι `True`.
- ```
  while συνθήκη:
    εντολές
  ```
- Αρχικά ελέγχεται η *συνθήκη* και αν είναι `True` εκτελούνται οι *εντολές*.
- Στη συνέχεια ελέγχεται ξανά η *συνθήκη* και αν πάλι είναι `True` εκτελούνται οι *εντολές* κ.ο.κ. Αν η *συνθήκη* γίνει `False`, σταματά η εκτέλεση της εντολής `while`.
  ```

Η εντολη `while`

- Η εντολή `while` εκτελεί μια σειρά εντολών όσο μια *συνθήκη*, που έχει ως αποτέλεσμα μια λογική τιμή, είναι `True`.
- ```
  while συνθήκη:
    εντολές
  ```
- Αρχικά ελέγχεται η *συνθήκη* και αν είναι `True` εκτελούνται οι *εντολές*.
- Στη συνέχεια ελέγχεται ξανά η *συνθήκη* και αν πάλι είναι `True` εκτελούνται οι *εντολές* κ.ο.κ. Αν η *συνθήκη* γίνει `False`, σταματά η εκτέλεση της εντολής `while`.
- Αν η *συνθήκη* δεν γίνει ποτέ `False`, τι θα συμβεί;
  ```

Παραδείγματα της εντολής `while`

Παραδείγματα της εντολής `while`

- Ας γράψουμε ένα απλό πρόγραμμα που θα τυπώνει τους αριθμούς από το 1 ως το 10.

Παραδείγματα της εντολής `while`

- Ας γράψουμε ένα απλό πρόγραμμα που θα τυπώνει τους αριθμούς από το 1 ως το 10.

```
i = 1
while i<=10:
  print(i)
  i = i + 1
```

Παραδείγματα της εντολής `while`

- Ας γράψουμε ένα απλό πρόγραμμα που θα τυπώνει τους αριθμούς από το 1 ως το 10.

```
i = 1
while i<=10:
  print(i)
  i = i + 1
```

- Ας γράψοουμε τώρα ένα πρόγραμμα που θα τυπώνει τους περιττούς αριθμούς από το 1 ως το 10.

Παραδείγματα της εντολής `while`

- Ας γράψουμε ένα απλό πρόγραμμα που θα τυπώνει τους αριθμούς από το 1 ως το 10.

```
i = 1
while i<=10:
  print(i)
  i = i + 1
```

- Ας γράψοουμε τώρα ένα πρόγραμμα που θα τυπώνει τους περιττούς αριθμούς από το 1 ως το 10.

```
i = 1
while i<=10:
  print(i)
  i = i + 2
```

Μετατροπή εντολής `for` σε εντολή `while`

Μετατροπή εντολής `for` σε εντολή `while`

- Ερώτημα: Μπορούμε να μεταρέψουμε μια εντολή `for` σε μία εντολή `while`;

Μετατροπή εντολής `for` σε εντολή `while`

- Ερώτημα: Μπορούμε να μεταρέψουμε μια εντολή `for` σε μία εντολή `while`;
- Η παρακάτω εντολή `for`

```
for μεταβλητή in range(αρχή,τέλος,βήμα):
  εντολές
```

Μετατροπή εντολής `for` σε εντολή `while`

- Ερώτημα: Μπορούμε να μεταρέψουμε μια εντολή `for` σε μία εντολή `while`;
- Η παρακάτω εντολή `for`

```
for μεταβλητή in range(αρχή,τέλος,βήμα):
  εντολές
```

- αντιστοιχεί στην παρακάτω εντολή `while`:

```
μεταβλητή = αρχή
while μεταβλητή < τέλος:
  εντολές
  μεταβλητή = μεταβλητή + βήμα
```

Η εντολη `continue`

Η εντολη `continue`

- Η εντολή `continue` σταματά την κανονική λειτουργία μια εντολής επανάληψης και την αναγκάζει να συνεχίσει από την αρχή.

Η εντολη `continue`

- Η εντολή `continue` σταματά την κανονική λειτουργία μια εντολής επανάληψης και την αναγκάζει να συνεχίσει από την αρχή.
- Ας δούμε ένα σχετικά απλό παράδειγμα χρήσης:

Η εντολη `continue`

- Η εντολή `continue` σταματά την κανονική λειτουργία μια εντολής επανάληψης και την αναγκάζει να συνεχίσει από την αρχή.
- Ας δούμε ένα σχετικά απλό παράδειγμα χρήσης:

```
for val in "string":
  if val == "i":
    continue
  print(val)
```

Η εντολη `continue`

- Η εντολή `continue` σταματά την κανονική λειτουργία μια εντολής επανάληψης και την αναγκάζει να συνεχίσει από την αρχή.
- Ας δούμε ένα σχετικά απλό παράδειγμα χρήσης:

```
for val in "string":
  if val == "i":
    continue
  print(val)
```

- Παρατηρούμε πως δεν υπάρχει η συνάρτηση `range` αλλά ένα string. Εδώ η μεταβλητή διατρέχει όλους τους χαρακτήρες που απαρτίζουν το string.

Η εντολη `continue` συνέχεια

Η εντολη `continue` συνέχεια

- Το αποτέλεσμα της εκτέλεσης του προηγούμενου κώδικα είναι:

Η εντολη `continue` συνέχεια

- Το αποτέλεσμα της εκτέλεσης του προηγούμενου κώδικα είναι:

```
s
t
r
n
g
```

Η εντολη `continue` συνέχεια

- Το αποτέλεσμα της εκτέλεσης του προηγούμενου κώδικα είναι:

```
s
t
r
n
g
```

- Ο λόγος είναι πως όταν η `val` λάβει την τιμή `"i"`, τότε εκτελείται η `continue` με αποτέλεσμα να αγνοηθεί για αυτή τη φορά η εκτέλεση της εντολής `print(val)`.

Η εντολη `break`

Η εντολη `break`

- Η εντολή `break` σταματά την λειτουργία μια εντολής επανάληψης στην οποία περιέχεται.

Η εντολη `break`

- Η εντολή `break` σταματά την λειτουργία μια εντολής επανάληψης στην οποία περιέχεται.
- Ας δούμε ένα σχετικά απλό παράδειγμα χρήσης:

Η εντολη `break`

- Η εντολή `break` σταματά την λειτουργία μια εντολής επανάληψης στην οποία περιέχεται.
- Ας δούμε ένα σχετικά απλό παράδειγμα χρήσης:

```
for val in "string":
  if val == "i":
    break
  print(val)
```

Η εντολη `break`

- Η εντολή `break` σταματά την λειτουργία μια εντολής επανάληψης στην οποία περιέχεται.
- Ας δούμε ένα σχετικά απλό παράδειγμα χρήσης:

```
for val in "string":
  if val == "i":
    break
  print(val)
```

- Το αποτέλεσμα της εκτέλεσης του κώδικα είναι:

Η εντολη `break`

- Η εντολή `break` σταματά την λειτουργία μια εντολής επανάληψης στην οποία περιέχεται.
- Ας δούμε ένα σχετικά απλό παράδειγμα χρήσης:

```
for val in "string":
  if val == "i":
    break
  print(val)
```

- Το αποτέλεσμα της εκτέλεσης του κώδικα είναι:

```
s
t
r
```

Η εντολη `break`

- Η εντολή `break` σταματά την λειτουργία μια εντολής επανάληψης στην οποία περιέχεται.
- Ας δούμε ένα σχετικά απλό παράδειγμα χρήσης:

```
for val in "string":
  if val == "i":
    break
  print(val)
```

- Το αποτέλεσμα της εκτέλεσης του κώδικα είναι:

```
s
t
r
```

- Γιατί;

Επίλογος

Επίλογος

- Περιγράψαμε τι είναι η Python·

Επίλογος

- Περιγράψαμε τι είναι η Python·
- γνωρίσαμε τους βασικούς τύπους τιμών της γλώσσας·

Επίλογος

- Περιγράψαμε τι είναι η Python·
- γνωρίσαμε τους βασικούς τύπους τιμών της γλώσσας·
- μάθαμε τις εντολές ανάθεσης, εισόδου και εξόδου·

Επίλογος

- Περιγράψαμε τι είναι η Python·
- γνωρίσαμε τους βασικούς τύπους τιμών της γλώσσας·
- μάθαμε τις εντολές ανάθεσης, εισόδου και εξόδου·
- επίσης μάθαμε τις βασικές εντολές ελέγχου της γλώσσας·

Επίλογος

- Περιγράψαμε τι είναι η Python·
- γνωρίσαμε τους βασικούς τύπους τιμών της γλώσσας·
- μάθαμε τις εντολές ανάθεσης, εισόδου και εξόδου·
- επίσης μάθαμε τις βασικές εντολές ελέγχου της γλώσσας·
- Σας ευχαριστώ για την προσοχή σας!