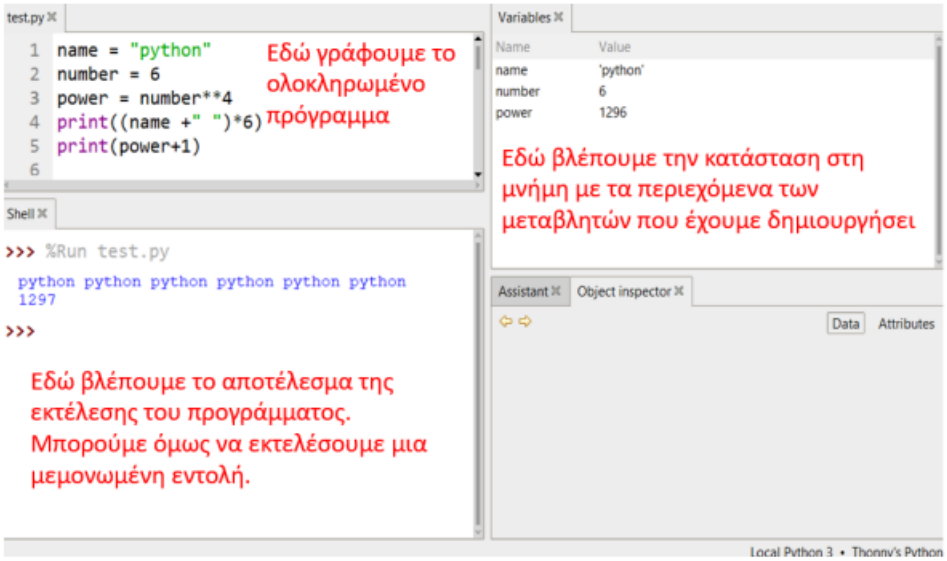
2.2.2 Το περιβάλλον προγραμματισμού Thonny

Το Thonny είναι ένα δωρεάν και ανοικτού κώδικα περιβάλλον προγραμματισμού για τη γλώσσα Python. Μπορείτε να το κατεβάσετε από τη διεύθυνση: https://thonny.org/.



Εικόνα 2.6. Το περιβάλλον προγραμματισμού Thonny

Εκτός από το παράθυρο του συντάκτη κειμένου που βρίσκεται πάνω αριστερά μπορούμε να δοκιμάσουμε και εντολές στο τερματικό του διερμηνευτή, αν θέλουμε να πειραματιστούμε και να δούμε τι λειτουργία επιτελούν. Εναλλακτικά περιβάλλοντα που μπορείτε να χρησιμοποιήσετε αντί για το Thonny είναι το IDLE (https://www.python.org/) και το Spyder (https://www.spyder-ide.org/).

2.3 Τύποι δεδομένων

Κάθε γλώσσα προγραμματισμού αποθηκεύει τα δεδομένα στη μνήμη σε διάφορες μορφές. Στο βιβλίο αυτό θα διαχωρίσουμε τα δεδομένα σε δυο βασικές κατηγορίες: τις λέξεις και τους αριθμούς. Οι λέξεις στην ορολογία της Πληροφορικής λέγονται αλφαριθμητικά, επειδή μπορούν να περιέχουν γράμματα και αριθμούς, όπως για παράδειγμα ο κωδικός πρόσβασής σας (password) στην ηλεκτρονική τάξη. Τα **αλφαριθμητικά** είναι μια ακολουθία από χαρακτήρες που μπορούν να είναι ψηφία, γράμματα ή σημεία στίξεως και βρίσκονται μέσα σε εισαγωγικά (quotes) (διπλά ή μονά). Οι **αριθμητικοί τύποι** στην Python είναι οι ακέραιοι (integer) (**int**) και οι πραγματικοί (**float**) αριθμοί. Για την υποδιαστολή στον προγραμματισμό χρησιμοποιείται η τελεία «.» και όχι το κόμμα.

Τα δεδομένα μπορεί να είναι αριθμοί, ονόματα ή αποτελέσματα λογικών αποφάσεων. Στην Python έχουμε τους εξής βασικούς τύπους:

Τύπος	Ονομασία	Παραδείγματα
Ακέραιοι	int	1, 0, –1, 496
Πραγματικοί	float	3.14159, 0.5, 3.0
Λογικές	bool	True, False
Αλφαριθμητικά	str	"SARS-CoV-2", "Ζάννειο Γυμνάσιο"
Λίστες	list	[6, 28, 496], ["Γη", "Σελήνη", "Ήλιος"]

Οι **λίστες** είναι ο σημαντικότερος σύνθετος τύπος της Python. Μια λίστα είναι μια ακολουθία από αντικείμενα οποιουδήποτε τύπου. Οι λίστες είναι δυναμικές δομές, δηλαδή μπορούμε να προσθέσουμε ή να αφαιρέσουμε στοιχεία αυξομειώνοντας το μέγεθός τους.
Η Python μας δίνει τη δυνατότητα να εκτελέσουμε διάφορες πράξεις στα δεδομένα, χρησιμοποιώντας τους αντίστοιχους τελεστές. Οι τελεστές (operators) είναι σύμβολα ή λέξεις για τη δημιουργία αριθμητικών και λογικών εκφράσεων. Οι βασικότερες κατηγορίες τελεστών είναι οι αριθμητικοί, οι συγκριτικοί και οι λογικοί. Υπάρχουν όμως και τελεστές για πράξεις σε αλφαριθμητικά δεδομένα και σε λίστες αντίστοιχοι με τους αριθμητικούς τελεστές.

Αριθμητικοί τελεστές: είναι τα σύμβολα που χρησιμοποιούμε για να κάνουμε μαθηματικές πράξεις. Ο υπολογισμός κάθε έκφραση στην οποία υπάρχουν αριθμητικοί τελεστές ακολουθεί μια αυστηρά καθορισμένη ιεραρχία πράξεων, που είναι:

1. Ύψωση σε δύναμη.
2. Πολλαπλασιασμός, διαίρεση.
3. Πρόσθεση, αφαίρεση.

Πρόσθεση	+
Αφαίρεση	–
Πολλαπλασιασμός	*
Διαίρεση	/
Πηλίκο Ακέραιας Διαίρεσης	//
Ύψωση σε δύναμη	**
Υπόλοιπο ακέραιας διαίρεσης	%

Αν θέλουμε να αλλάξουμε την ιεραρχία των πράξεων, μπορούμε να χρησιμοποιήσουμε παρενθέσεις. Για παράδειγμα, στην έκφραση (200+1)*10 θα εκτελεστεί πρώτα η πρόσθεση μέσα στην παρένθεση και μετά το αποτέλεσμα θα πολλαπλασιαστεί επί 10, που μας κάνει 2010, σε αντίθεση με την έκφραση 200+1*10, στην οποία πρώτα θα γίνει ο πολλαπλασιασμός και μετά η πρόσθεση, άρα θα πάρουμε 210.

Οι **Σχεσιακοί** (ή συγκριτικοί) τελεστές χρησιμοποιούνται για τη σύγκριση εκφράσεων. Αν η σχέση ισχύει το αποτέλεσμα είναι Αληθής (True) αλλιώς Ψευδής (False). Μπορούμε να τις αντιμετωπίζουμε ως ερωτήσεις που απαντώνται με ΝΑΙ ή ΟΧΙ.	Μικρότερο από	**<**
	Μικρότερο ή ίσο από	**<=**
	Μεγαλύτερο από	**>**
	Μεγαλύτερο ή ίσο από	**>=**
	Ίσο με	**==**
	Διάφορο από	**!=**

```
>>> 496==496  >>> 12<11 and 23>10
True          False
>>> 12!=13    >>> 12<11 or 23>10   >>> not(56<=12)
True          True                 True
```

Οι **Λογικοί** τελεστές χρησιμοποιούνται όταν θέλουμε να σχηματίσουμε σύνθετες συνθήκες. Η σύζευξη λογικών εκφράσεων είναι Αληθής, αν και μόνον αν είναι αληθείς όλες οι εκφράσεις, ενώ η διάζευξη είναι αληθής, αν συμμετέχει σε αυτήν τουλάχιστον μια αληθής έκφραση.	Άρνηση (ΟΧΙ)	**not**
	Σύζευξη (ΚΑΙ)	**and**
	Διάζευξη (Η)	**or**

Μπορούμε να δοκιμάσουμε εντολές ή να υπολογίσουμε εκφράσεις χρησιμοποιώντας τον διερμηνευτή της Python, ο οποίος μεταφράζει σε γλώσσα μηχανής και εκτελεί μια μεμονωμένη εντολή χωρίς να χρειαστεί να αναπτύξουμε ολόκληρο πρόγραμμα. Ο τελεστής (συνάρτηση) str μετατρέπει ένα αντικείμενο άλλου τύπου σε αλφαριθμητικό. Ο τελεστής της πρόσθεσης «+» όταν εφαρμόζεται σε αλφαριθμητικά, εκτελεί συνένωση ενώ αυτός του πολλαπλασιασμού «*» παράγει τη συμβολοσειρά επαναλαμβανόμενη τόσες φορές, όσες είναι ο αριθμός. Ο τελεστής int μετατρέπει ένα αντικείμενο σε ακέραιο αριθμό, εάν αυτό είναι εφικτό.

```
>>> 10+10
20
>>> "10"+"10"
'1010'
>>> 5*"10"
'1010101010'
>>> 5*10
50
>>> 5*int("10")
50
>>> 5*str(10)
'1010101010'
```

```
>>> number = input("Δώσε έναν αριθμό = ")
 Δώσε έναν αριθμό = 28
>>> print(number+number+number)
 282828
>>> print(3*number)
 282828
>>> number = int( input("Δώσε έναν αριθμό = ") )
 Δώσε έναν αριθμό = 28
>>> print(number+number+number)
 84
>>> print(3*number)
 84
```

Οι τελεστές μετατροπής τύπων int, float και str χρειάζονται κατά την εισαγωγή δεδομένων από τον χρήστη ή από άλλη πηγή, για παράδειγμα από κάποιο αρχείο. Ακόμα και αριθμό να δώσει ο χρήστης, η Python θα το εκλάβει ως αλφαριθμητικό, οπότε θα χρειαστεί να το μετατρέψουμε στην κατάλληλη μορφή. Στο παραπάνω παράδειγμα ο χρήστης δίνει τον αριθμό 28, τον οποίο διαβάζει η εντολή input. Στην πρώτη περίπτωση αντιμετωπίζεται σαν αλφαριθμητικό και στη δεύτερη σαν ακέραιος αριθμός.

Δραστηριότητα 1

Μεταβείτε στο κέλυφος (shell) του Thonny και δώστε τις παρακάτω εκφράσεις:

```
>>> 2**10   >>> 2**100   >>> 2**1000   >>> 2**10000
```

Δραστηριότητα 2

Μεταβείτε στο κέλυφος (shell) του Thonny και δώστε τις παρακάτω εκφράσεις:

```
>>> 5*("Grace" + "Hopper" + " , ")     >>> 5*"Grace" + 5*"Hopper" + 5*" , "     >>> "Alan Turing"[5:10]
```

Στη συνέχεια μεταβείτε στο ChatGPT ή στο Gemini και δώστε την προτροπή:
Μπορείς να μου εξηγήσεις τη λειτουργία που επιτελεί η έκφραση "python"[2:4]

Δραστηριότητα 3

Μεταβείτε στο κέλυφος (shell) του Thonny και δώστε τις παρακάτω εκφράσεις:

```
>>> len(str(8128))   >>> len(str(496))   >>> len(str(10**20))   >>> len(str(2**100))   >>> str(len(5))
```

Τι κάνει η συνάρτηση len και τι η str; Τι πετυχαίνουμε με τη συνδυασμένη χρήση τους;
Στη συνέχεια μεταβείτε στο ChatGPT ή στο Gemini και δώστε την προτροπή:
Ποια λειτουργία επιτελεί η len(str(number));