Εισαγωγή στον Προγραμματισμό με Python

Δοκιμάστε να αντιγράψετε και να επικολλήσετε τον κώδικα σε ένα "Code" κελί του Jupyter Notebook, εκτελέστε το, και πειραματιστείτε με τις τιμές για να δείτε τα αποτελέσματα.

🇬🇧 For Erasmus Students, I recommend the w3schools Python Tutorial (in English): https://www.w3schools.com/python

Μεταβλητές και Τύποι Δεδομένων

Στην Python, οι μεταβλητές είναι "δοχεία" που αποθηκεύουν δεδομένα. Κάθε μεταβλητή έχει έναν τύπο δεδομένων που καθορίζει τι είδους πληροφορίες μπορεί να αποθηκεύσει.

Βασικοί τύποι δεδομένων:

- `int` : Ακέραιοι αριθμοί
- `float` : Δεκαδικοί αριθμοί
- `str` : Συμβολοσειρές (κείμενο)
- `bool` : Λογικές τιμές (True/False)

Παραδείγματα:

```
age = 20                # int
height = 1.75           # float
name = "Μαρία"          # str
is_student = True       # bool

print(type(age))        # <class 'int'>
print(type(height))     # <class 'float'>
print(type(name))       # <class 'str'>
print(type(is_student)) # <class 'bool'>
```

In [ ]:

Πράξεις

Η Python υποστηρίζει διάφορες αριθμητικές πράξεις:

```
a = 10
b = 3

print(a + b)  # Πρόσθεση: 13
print(a - b)  # Αφαίρεση: 7
print(a * b)  # Πολλαπλασιασμός: 30
print(a / b)  # Διαίρεση: 3.3333333333333335
print(a // b) # Ακέραια διαίρεση: 3
print(a % b)  # Υπόλοιπο: 1
print(a ** b) # Δύναμη: 1000
```

In [ ]:

f-strings

Τα f-strings επιτρέπουν την εύκολη ενσωμάτωση μεταβλητών μέσα σε συμβολοσειρές:

```
name = "Γιάννης"
age = 25

message = f"Ο {name} είναι {age} ετών."
print(message)  # Ο Γιάννης είναι 25 ετών.

pi = 3.14159
print(f"Το π είναι περίπου {pi:.2f}")  # Το π είναι περίπου 3.14
```

In [ ]:

Λογικές Συνθήκες Σύγκρισης

Οι λογικοί τελεστές σύγκρισης χρησιμοποιούνται για τη σύγκριση τιμών:

- `==` : ίσο με
- `!=` : διάφορο του
- `<` : μικρότερο από
- `>` : μεγαλύτερο από
- `<=` : μικρότερο ή ίσο με
- `>=` : μεγαλύτερο ή ίσο με

```
x = 5
y = 10

print(x == y)  # False
print(x != y)  # True
print(x < y)   # True
print(x > y)   # False
print(x <= 5)  # True
print(y >= 10) # True
```

In [ ]:

Σύνταξη Python και Στοίχιση (Indentation)

Η Python χρησιμοποιεί στοίχιση για να ορίσει μπλοκ κώδικα. Αυτό είναι ένα από τα πιο χαρακτηριστικά και σημαντικά στοιχεία της σύνταξης της Python, που την διαφοροποιεί από πολλές άλλες γλώσσες προγραμματισμού.

Τι είναι η Στοίχιση;

Η στοίχιση αναφέρεται στα κενά στην αρχή κάθε γραμμής κώδικα. Στην Python, αυτά τα κενά έχουν σημασία και καθορίζουν τη δομή του προγράμματος.

Πώς Λειτουργεί η Στοίχιση:

1. **Ορισμός Μπλοκ Κώδικα**: Κάθε μπλοκ κώδικα (π.χ., το σώμα μιας συνάρτησης, μιας δομής ελέγχου) ξεκινά με αυξημένη στοίχιση.

2. **Συνέπεια**: Όλες οι γραμμές σε ένα μπλοκ πρέπει να έχουν την ίδια στοίχιση.

3. **Τέλος Μπλοκ**: Ένα μπλοκ τελειώνει όταν η στοίχιση επιστρέφει στο προηγούμενο επίπεδο.

4. **Αριθμός Κενών**: Συνήθως χρησιμοποιούνται 4 κενά για κάθε επίπεδο στοίχισης, αν και μπορείτε να χρησιμοποιήσετε οποιονδήποτε σταθερό αριθμό κενών.

Παραδείγματα:

1. **Συναρτήσεις**:

```
def greet(name):
    print(f"Γεια σου, {name}!")  # Στοιχισμένο μπλοκ
    print("Καλώς ήρθες!")        # Ίδια στοίχιση, ίδιο μπλοκ
print("Τέλος συνάρτησης")       # Χωρίς στοίχιση, εκτός της συνάρτησης

greet("Μαρία")
```

2. **Δομές Ελέγχου**:

```
age = 20
if age >= 18:
    print("Ενήλικας")        # Στοιχισμένο μπλοκ για το if
    print("Μπορείς να ψηφίσεις")
else:
    print("Ανήλικος")        # Στοιχισμένο μπλοκ για το else
    print("Δεν μπορείς να ψηφίσεις")
print("Αυτό εκτελείται πάντα")  # Εκτός των μπλοκ if/else
```

3. **Εμφωλευμένα Μπλοκ**:

```
def check_grade(grade):
    if grade >= 90:
        print("Άριστα!")
        if grade == 100:
            print("Τέλειο σκορ!")  # Διπλά στοιχισμένο
    elif grade >= 70:
        print("Καλά")
    else:
        print("Χρειάζεται βελτίωση")

check_grade(95)
```

Σημαντικά Σημεία:

- **Ευαναγνωσιμότητα**: Η στοίχιση κάνει τον κώδικα πιο ευανάγνωστο, καθώς η δομή του είναι οπτικά εμφανής.
- **Λιγότερα Λάθη**: Μειώνει την πιθανότητα λαθών που σχετίζονται με τη δομή του κώδικα.
- **Συνέπεια**: Επιβάλλει μια συνεπή μορφοποίηση σε όλο τον κώδικα.
- **Προσοχή στα Λάθη**: Λανθασμένη στοίχιση μπορεί να οδηγήσει σε σφάλματα `IndentationError` ή απρόσμενη συμπεριφορά του κώδικα.

Συμβουλές:

- Να είστε συνεπείς στη χρήση είτε κενών είτε tabs (προτιμώνται τα κενά).
- Ελέγχετε προσεκτικά τη στοίχιση όταν αντιγράφετε και επικολλάτε κώδικα.

Η κατανόηση και η σωστή χρήση της στοίχισης είναι βασική για τον αποτελεσματικό προγραμματισμό σε Python. Είναι ένα από τα πρώτα πράγματα που πρέπει να εξοικειωθούν οι νέοι προγραμματιστές Python.

In [1]:
```
def check_grade(grade):
    if grade >= 90:
        print("Άριστα!")
        if grade == 100:
            print("Τέλειο σκορ!")  # Διπλά στοιχισμένο
    elif grade >= 70:
        print("Καλά")
    else:
        print("Χρειάζεται βελτίωση")

check_grade(95)
```

```
Άριστα!
```

Δομές Ελέγχου: if/elif/else

Οι δομές ελέγχου επιτρέπουν την εκτέλεση διαφορετικών τμημάτων κώδικα βάσει συνθηκών:

```
grade = 85

if grade >= 90:
    print("Άριστα!")
elif grade >= 80:
    print("Πολύ καλά!")
elif grade >= 70:
    print("Καλά.")
elif grade >= 60:
    print("Μέτρια.")
else:
    print("Χρειάζεται βελτίωση.")

# Αποτέλεσμα: Πολύ καλά!
```

Αυτές οι δομές μπορούν να συνδυαστούν με λογικούς τελεστές για πιο σύνθετες συνθήκες:

```
age = 25
is_student = True
is_greek_citizen = True

if age <= 25 and is_student:
    print("Δωρεάν είσοδος στον αρχαιολογικό χώρο.")
elif is_greek_citizen:
    if age < 18 or age >= 65:
        print("Μειωμένο εισιτήριο εισόδου.")
    else:
        print("Κανονικό εισιτήριο εισόδου για Έλληνες πολίτες.")
else:
    print("Κανονικό εισιτήριο εισόδου για μη Έλληνες επισκέπτες.")

# Αποτέλεσμα: Δωρεάν είσοδος στον αρχαιολογικό χώρο.
```

Σε αυτό το παράδειγμα:

1. Ελέγχουμε πρώτα αν ο επισκέπτης είναι φοιτητής ηλικίας έως 25 ετών, οπότε δικαιούται δωρεάν είσοδο.
2. Αν όχι, ελέγχουμε αν είναι Έλληνας πολίτης.
   - Αν είναι Έλληνας και είτε ανήλικος (< 18) είτε άνω των 65, δικαιούται μειωμένο εισιτήριο.
   - Διαφορετικά, πληρώνει το κανονικό εισιτήριο για Έλληνες πολίτες.
3. Αν δεν είναι Έλληνας πολίτης, πληρώνει το κανονικό εισιτήριο για μη Έλληνες επισκέπτες.

Αυτό το παράδειγμα δείχνει πώς μπορούμε να χρησιμοποιήσουμε πολλαπλές συνθήκες και εμφωλευμένες δομές ελέγχου για να χειριστούμε διαφορετικές περιπτώσεις, όπως συμβαίνει συχνά σε πραγματικά σενάρια εφαρμογών.

In [ ]: