الوحدة 5: هياكل البيانات (Data Structures)

هياكل البيانات (Data Structures) هي طرق لتنظيم وتخزين البيانات في الذاكرة بحيث يمكن استخدامها بكفاءة. بايثون توفر مجموعة من هياكل البيانات المدمجة التي تُستخدم بشكل واسع في البرمجة اليومية.

من أهم هذه البُنى:

- القوائم(List)
- السلاسل النصية(String)
- الصفوف(Tuple)
- القواميس(Dictionary)
- المجموعات(Set)

أ. القوائم(Lists)

القائمة (List) هي بنية بيانات قابلة للتغيير (Mutable) تُستخدم لتخزين عدة عناصر في متغير واحد. يمكن أن تحتوي القائمة على عناصر من أنواع مختلفة.

إنشاء قائمة

```
numbers = [1, 2, 3, 4, 5]

names = ["Ali", "Sara", "Omar"]

mixed = [1, "Python", 3.14, True]
```

الوصول إلى العناصر

يتم الوصول إلى العناصر باستخدام الفهرس (Index) الذي يبدأ من 0:

```
print(names[0])   # Ali

print(numbers[2]) # 3
```

تعديل العناصر

```
names[1] = "Amina"
```

إضافة عناصر

```
names.append("Hassan")        # إضافة في النهاية

names.insert(1, "Youssef")    # إضافة في موقع محدد
```

حذف عناصر

```
names.remove("Ali")   # حذف عنصر بالقيمة

del names[0]          # حذف عنصر بالفهرس
```

التكرار داخل قائمة

```
for name in names:

    print(name)
```

مثال: حساب معدل درجات

```
grades = [15, 18, 12, 10]

average = sum(grades) / len(grades)

print("Average:", average)
```

ب. 5.3السلاسل النصية(Strings)

السلسلة النصية (String) هي تسلسل من الحروف (Characters) تُستخدم لتخزين النصوص.

السلاسل غير قابلة للتغيير.(Immutable)

```
message = "Hello, Python!"
```

الوصول إلى الحروف

```
print(message[0])   # H

print(message[-1])  # !
```

تقطيع السلاسل(Slicing)

```
print(message[0:5])  # Hello
```

دوال شائعة

```
print(len(message))          # طول السلسلة

print(message.upper())       # أحرف كبيرة

print(message.lower())       # أحرف صغيرة

print(message.replace("Hello", "Hi"))  # استبدال
```

التكرار داخل سلسلة

```
for char in message:

    print(char)
```

مثال: تحليل جملة

```
sentence = "Python is powerful"

words = sentence.split()

print("Number of words:", len(words))
```

ج. الصفوف(Tuples)

الصف (Tuple) هو بنية بيانات تشبه القائمة (List) لكنها **غير قابلة للتغيير(Immutable)** ، أي لا يمكن تعديل عناصرها بعد إنشائها.

يُستخدم الـ Tuple عندما نريد تخزين بيانات ثابتة.

إنشاء Tuple

```
numbers = (1, 2, 3, 4, 5)

person = ("Ali", 20, "Student")
```

الوصول إلى العناصر

```
print(person[0])  # Ali

print(numbers[2]) # 3
```

التكرار داخل Tuple

```
for item in person:

print(item)
```

ملاحظة حول عدم القابلية للتغيير

```
person[0] = "Sara"  # خطأ: لا يمكن التعديل
```

للتعديل غير المباشر، يجب التحويل إلى قائمة:

```
person_list = list(person)

person_list[0] = "Sara"

person = tuple(person_list)
```

استخدامات الـTuple

- تخزين بيانات ثابتة لا تتغير.
- إرجاع عدة قيم من دالة واحدة.
- تحسين الأداء عند التعامل مع بيانات ثابتة.

مثال: إرجاع أكثر من قيمة من دالة

```
def student_info():
    return ("Ali", 20, 16.5)


info = student_info()
print("Name:", info[0])
print("Age:", info[1])
print("Grade:", info[2])
```

د. القواميس(Dictionaries)

القاموس (Dictionary) هو بنية بيانات تُخزن القيم في شكل أزواج "مفتاح:قيمة.(Key: Value) "
المفاتيح يجب أن تكون فريدة ولا تتكرر.

إنشاء قاموس

```
student = {
    "name": "Ali",
    "age": 20,
    "grade": 15.5
```

```
}
```

الوصول إلى القيم

```
print(student["name"])   # Ali
```

تعديل القيم

```
student["grade"] = 16.0
```

إضافة وحذف مفاتيح

```
student["major"] = "Computer Science"

del student["age"]
```

التكرار داخل قاموس

```
for key, value in student.items():

    print(key, ":", value)
```

مثال : إدارة درجات طلاب

```
students = {

    "Ali": 15.5,

    "Sara": 17.0,

    "Omar": 13.5

}



for name, grade in students.items():
```

```
print(name, ":", grade)
```

هـ. المجموعات(Sets)

المجموعة (Set) هي بنية بيانات غير مرتبة (Unordered) تحتوي على عناصر فريدة (Unique) ولا تسمح بالتكرار.

إنشاء مجموعة

```
fruits = {"apple", "banana", "orange"}
```

إضافة وحذف عناصر

```
fruits.add("mango")

fruits.remove("apple")
```

العمليات على المجموعات

```
A = {1, 2, 3}

B = {3, 4, 5}



print(A.union(B))        # اتحاد

print(A.intersection(B)) # تقاطع

print(A.difference(B))   # فرق
```

التكرار داخل مجموعة

```
for item in fruits:
```

```
    print(item)
```

مثال واقعي – إزالة التكرار

```
numbers = [1, 2, 2, 3, 4, 4, 5]

unique_numbers = set(numbers)

print(unique_numbers)
```

و. التكرار داخل المجموعات(Iteration in Data Structures)

يمكن استخدام حلقة for للتكرار عبر جميع أنواع المجموعات:

```
students = ["Ali", "Sara", "Omar"]

for student in students:

    print(student)
```

مثال 1: إدارة بيانات طلاب

```
students = {

    "Ali": 15.5,

    "Sara": 17.0,

    "Omar": 13.5

}

for name, grade in students.items():

    if grade >= 15:
```

```
        print(name, "Passed")
    else:
        print(name, "Failed")
```

مثال 2: إدارة مخزون بضائع

```
inventory = {
    "Apples": 50,
    "Bananas": 30,
    "Oranges": 20
}

for item, quantity in inventory.items():
    print(item, ":", quantity)

inventory["Apples"] += 10
print("Updated Apples:", inventory["Apples"])
```

ز. مقارنة بين هياكل البيانات في بايثون(Comparison of Data Structures)

البنية (Structure)	القابلية للتغيير (Mutable)	القابلية للتكرار (Iterable)	السماح بالتكرار (Duplicates)	الفهرسة (Indexing)	الترتيب (Ordered)	الصيغة (Syntax)	أمثلة الاستخدام
List (قائمة)	نعم	نعم	نعم	نعم	نعم	[1, 2, 3]	تخزين بيانات متغيرة مثل درجات الطلبة
String (سلسلة)	لا	نعم	نعم	نعم	نعم	"Hello"	تخزين النصوص ومعالجتها
Tuple (صف)	لا	نعم	نعم	نعم	نعم	(1, 2, 3)	تخزين بيانات ثابتة مثل إعدادات البرنامج
Dictionary (قاموس)	نعم	نعم	لا (المفاتيح)	لا	نعم من Python 3.7+	{"key": "value"}	تخزين بيانات في شكل مفتاح وقيمة مثل ملفات الإعدادات
Set (مجموعة)	نعم	نعم	لا	لا	لا	{1, 2, 3}	إزالة التكرارات والعمليات الرياضية على المجموعات

ملاحظات هامة:

- **List** هي أكثر أنواع المجموعات استخدامًا في المشاريع العامة نظرًا لمرونتها.
- **Tuple** أسرع من **List** وتُستخدم عندما تكون البيانات ثابتة ولا تحتاج إلى تعديل.
- **Dictionary** هو الأفضل في تخزين البيانات المرتبطة (Key:Value) مثل قواعد البيانات المصغّرة.
- **Set** ممتاز عند الحاجة إلى عناصر فريدة أو إجراء عمليات مثل الاتحاد (Union) والتقاطع (Intersection).
- **String** ليست "مجموعة" بالمعنى الهيكلي، لكنها تُعامل كسلسلة قابلة للتكرار وتُستخدم بكثرة