

## 📌 تعريف المتغيرات

يُكتب اسم المتغير، `let` تسمح لك بإنشاء متغيرات باستخدام الكلمة المفتاحية `S` لغة .نوعه، ثم قيمته.

### ✅ الشكل العام:

```
```s
let المتغير: النوع = القيمة;
```
```

### 🔧 أمثلة عملية:

```
```s
let age: int = 25;
let pi: float = 3.14;
let isReady: bool = true;
let name: string = "So2_gemel";
```
```

## 🧠 أنواع البيانات المدعومة

| النوع   | الوصف   | مثال                        |
|---|---|-----------------------------|
| <code>let x: int = 5;</code>  | عدد صحيح                                      | <code>int</code>            |
| <code>let y: float = 1.5;</code>  | عدد عشري                                      | <code>float</code>          |
| <code>let ok: bool = false;</code>                                      | <code>true</code> أو <code>false</code> منطقي | <code>bool</code>           |
| <code>let msg: string = "Hello";</code>                                 | نص  | <code>string</code>         |
| <code>let nums: list&lt;int&gt; = [1, 2, 3];</code>                     | T قائمة تحتوي عناصر من نوع                    | <code>list&lt;T&gt;</code>  |
| <code>let dict: map&lt;string,int&gt; =</code><br><code>{"a":1};</code> | خريطة مفتاح ↔ قيمة                            | <code>map&lt;K,V&gt;</code> |

## 📢 الطباعة

لعرض النصوص على الشاشة `print()` تستخدم.

### 📌 أمثلة:

```
```s
print("مرحبًا بلغتي الجديدة");
```
```

### 🌟 دمج نص + متغير:

```
```s
let name: string = "So2_gemel";
print("أهلاً يا " + name);
```
```

## 🔗 تمرين عملي

:اكتب برنامجًا يعرف الاسم والعمر، ويطبع رسالة ترحيب

```
```s
func main() {
    let name: string = "So2_gemel";
    let age: int = 23;
}
```

```
    print("مرحبًا " + name);  
    print("عمركَ هو " + age);  
}
```

## S الفصل 3: التحكم بالتدفق - الشروط والحلقات في لغة

---

🌀 ما معنى التحكم بالتدفق؟

التحكم بالتدفق هو الطريقة التي يقرر بها البرنامج أي جزء من الكود يُنفذ، وأي جزء هذا يتم باستخدام S، يُتجاوز، ومتى تُعاد بعض الأوامر بشكل دوري. في لغة

- الشروط: if / else
- الحلقات: for / while

---

🧠 if / else أولاً: الشروط

✅ الشكل العام:

```
```s
if الشرط {
    // ينفذ هذا الكود إذا كان الشرط صحيح
} else {
    // ينفذ هذا الكود إذا كان الشرط خاطئ
}
```
```

📌 مثال تطبيقي:

```
```s
let temp: int = 34;

if temp > 30 {
    print("الجو حار");
} else {
    print("الجو معتدل");
}
```
```

---

🔄 ثانيًا: الحلقات

✅ حلقة for

.تستخدم عند معرفة عدد التكرارات مسبقًا.

```
```s
for i in 0..5 {
    print("التكرار رقم: " + i);
}
```
```

المدى 0..5 يعني من 0 إلى 4

---

✅ حلقة while

.تستخدم عندما لا تعرف عدد التكرارات، لكنك تضع شرطًا للتوقف.

```
```s
let count: int = 0;

while count < 3 {
```

```
    print("جاري التكرار " + count);
    count = count + 1;
}
```

---

استخدام الشروط داخل دوال

```
```s
func checkGrade(mark: int): string {
    if mark >= 90 {
        return "امتياز";
    } else {
        return "حاول أكثر";
    }
}
```

---

تمرين عملي

:اكتب برنامجًا يسأل عن العمر ويحدد التصنيف

```
```s
func main() {
    let age: int = 19;

    if age < 13 {
        print("أنت طفل");
    } else {
        if age < 20 {
            print("أنت مرافق");
        } else {
            print("أنت بالغ");
        }
    }
}
```

---

## الأصناف، البناء، والوظائف داخل - S++ الفصل 4: البرمجة الكائنية باستخدام الكائنات

---

### 🎯 ما هي البرمجة الكائنية؟

البرمجة الكائنية تعني أنك تصمم "كائنات" تمثل أشياء في التطبيق، مثل مستخدم، منتج، ... نافذة، ملف. كل كائن له بيانات (خصائص) وله وظائف (سلوك).

---

### 🏠 (Class) إنشاء صنف

class. ننشئ صنف باستخدام الكلمة S++، في

### ✅ الشكل العام:

```
```s++
class اسم_الصنف {
    // خصائص
    // دوال
}
```
```

### 📌 مثال عملي:

```
```s++
class Person {
    let name: string;
    let age: int;

    func init(n: string, a: int) {
        self.name = n;
        self.age = a;
    }

    func greet(): string {
        return "أهلاً يا " + self.name;
    }
}
```
```

---

### 🔍 شرح عناصر الصنف

| العنصر | الوصف                                      |
|--------|--|
| class  | إنشاء صنف                                  |
| let    | تعريف خاصية داخل الصنف                     |
| func   | تعريف دالة داخل الصنف                      |
| init   | دالة البناء، تُنفذ تلقائيًا عند إنشاء كائن |
| self   | تشير إلى الكائن الحالي                     |

---

### 👉 (Object) إنشاء كائن

:(أو الشكل الذي تراه مناسبًا لاحقًا) new بعد تعريف الصنف، ننشئ كائن باستخدام

```
```s++
let p: Person = new Person("So2_gemel", 23);
```
```

```
print(p.greet());
```

```
\\\
```

```
---
```

✂ تمرين عملي

:اكتب صنفًا يمثل سيارة

```
```s++
```

```
class Car {
    let brand: string;
    let year: int;

    func init(b: string, y: int) {
        self.brand = b;
        self.year = y;
    }

    func info(): string {
        return "سيارة " + self.brand + " من سنة " + self.year;
    }
}

func main() {
    let c: Car = new Car("Toyota", 2020);
    print(c.info());
}
```
```

```
---
```

## 📄 S/S++ الفصل 5: بناء مشاريع عملية باستخدام لغة

---

### 🎯 لماذا المشاريع العملية؟

تطبيق المعرفة النظرية في مشاريع حقيقية هو الوسيلة الأقوى لتثبيت المفاهيم، واكتشاف الثغرات، وتحسين التصميم. تم تصميمها لتكون مناسبة لتطبيقات متنوعة: أدوات صغيرة، برامج سطح S/S++ لغة المكتب، أنظمة تعليمية، وحتى واجهات رسومية لاحقًا.

---

### 🧮 المشروع 1: آلة حاسبة بسيطة

#### ✅ الفكرة:

تطبيق يحسب العمليات الرياضية الأربعة بناءً على المُدخلات.

#### 📁 الكود:

```
```s
func add(a: int, b: int): int {
    return a + b;
}

func sub(a: int, b: int): int {
    return a - b;
}

func main() {
    let x: int = 10;
    let y: int = 5;

    print("مجموع: " + add(x, y));
    print("فرق: " + sub(x, y));
}
```
```

---

### 📄 (Text Logger) المشروع 2: مدونة نصية

#### ✅ الفكرة:

برنامج يحتفظ بقائمة نصوص ويعرضها لاحقًا.

#### 📁 الكود:

```
```s++
class Logger {
    let logs: list<string>;

    func init() {
        self.logs = [];
    }

    func add(msg: string) {
        self.logs.push(msg);
    }

    func show() {
        for i in 0..len(self.logs) {
            print "[" + i + "]: " + self.logs[i];
        }
    }
}
```

```

    }
}

func main() {
    let log: Logger = new Logger();
    log.add("بدأ البرنامج");
    log.add("تمت إضافة سجل جديد");
    log.show();
}

```

---

### 🧠 المشروع 3: أداة تحليل النصوص

#### ✅ الفكرة:

. تحليل عدد الكلمات والحروف في نص مدخل

```

```s
func countWords(text: string): int {
    return len(text.split(" "));
}

func countChars(text: string): int {
    return len(text);
}

func main() {
    let msg: string = "تبني مستقبل البرمجة S لغة";
    print("كلمات: " + countWords(msg));
    print("أحرف: " + countChars(msg));
}

```

---

#### 🚀 نصائح لتنظيم المشاريع:

- examples/ ضع كل مشروع في مجلد داخل
- ast\_samples/ لكل مثال في AST أضف ملف
- docs/ch5\_projects.md أرفق شرح داخل

---



دوال جاهزة، توثيقها، وربطها مع - core.slib الفصل 6: بناء مكتبة قياسية Interpreter

---

لماذا نحتاج مكتبة قياسية؟

كل لغة برمجة قوية تملك مجموعة دوال وأدوات جاهزة تُسهّل على المطور بناء المشاريع دون الحاجة لكتابة كل شيء من الصفر. تكون مرجعية أساسية لكل البرامج، core.slib تُنشئ مكتبة رسمية تدعى S/S++ في لغة.

---

إنشاء ملف المكتبة

: بهذا المسار stdlib/ ضع الملف داخل مجلد

```
```txt
stdlib/core.slib
```
```

S. أو أي تنسيق خاص بلغة JSON أو DSL صيغة المكتبة يمكن أن تكون شبه

---

الدوال القياسية المقترحة

| الدالة  | الوصف                           | التوقيع         |
|---|---------------------------------|-----------------|
| func print(text: string)                            | طباعة إلى الشاشة                | print(text)     |
| func input(prompt: string): string                  | قراءة من المستخدم               | input(prompt)   |
| func len(item: list<T>): int                        | حساب الطول لأي عنصر قابل للقياس | len(item)       |
| func type(var: any): string                         | معرفة نوع متغير                 | type(var)       |
| func add(a: int, b: int): int                       | جمع رقمين                       | add(a,b)        |
| func split(text: string, sep: string): list<string> | تقسيم النص                      | split(text,sep) |

ويمكن لاحقًا ترجمتها إلى تنفيذ فعلي داخل الـ core.slib هذه الدوال تُكتب داخل Interpreter

---

مثال مكتبة نصية

```
```s
func print(text: string);
func input(prompt: string): string;
func len(item: list<any>): int;
func add(a: int, b: int): int;
func type(x: any): string;
```
```

---

بها؟ Interpreter كيف يرتبط الـ

:أولاً Interpreter يبحث الـ AST عند تنفيذ الـ

1. (let, print مثل) هل الأمر يُنفذ داخليًا
2. core.slib هل الأمر موجود في
3. إذا وُجد، يتم استدعاؤه من المكتبة، أو ترجمة الكود إلى عملية فعلية

✔ مثال :

```
```s
print("أهلاً");
```
```

الـ Interpreter في الـ

```
```python
if node["type"] == "PrintStatement":
    # ينفذ الكود مباشرة
elif node["type"] == "Call":
    # عن الدالة ثم ينفذها core.slib يبحث في
...

---
```

بيئة تعليمية، محرر ذكي، وتنفيذ حيّ - S/S++ الفصل 7: تصميم واجهة رسومية للغة للكود

---

لماذا واجهة رسومية؟

في عصر البرمجة الحديثة، التفاعل مع اللغة لا يقتصر على الطرفية والنصوص. واجهة رسومية قوية يمكن أن:

- تختصر الحواجز التقنية للمبتدئين
- تسرّع عملية التطوير والتجربة
- تربط التعلم النظري بتنفيذ عملي حيّ
- تصبح بيئة موحدة لتطوير، تحليل، توثيق، وتنفيذ الكوادر

---

الذكي S مكونات محرر

المكوّن	الوظيفة
S/S++ محرر نصي	كتابة الشيفرة بلغة
s_parser.py عبر AST زر "تحليل الكود"	لتحويل الشيفرة إلى
interpreter.py باستخدام AST زر "تنفيذ الكود"	لتشغيل
نافذة النتائج	لعرض المخرجات أو الأخطاء
examples/ قائمة المشاريع	لاستعراض وتشغيل الأمثلة من
docs/*.md تبويب التوثيق	عرض محتوى الفصل التعليمي الحالي من
دعم الحفظ والفتح	حفظ أكواد واختبارها لاحقًا بسهولة

---

فلسفة التصميم: واجهة تعليمية تنفيذية

> تجربة واحدة، تجمع  
> الكتابة + التحليل + التنفيذ + التعلم + التوثيق

---

الخطوات المعمارية لبناء الواجهة

GUI المرحلة 1: اختيار مكتبة

- Python سريعة، مدمجة مع Tkinter: مبدئيًا
- لدعم موسّع ومرونة واجهات PyQt / Electron: لاحقًا

---

المرحلة 2: بناء الواجهة الأولية

```
python
import tkinter as tk
from tkinter import filedialog, scrolledtext

def run_code():
    code = editor.get("1.0", tk.END)
    with open("temp.s", "w", encoding="utf-8") as f:
        f.write(code)
    # تشغيله ← يتم ربطها لاحقًا، تحليل الكود، إنتاج
```

```

root = tk.Tk()
root.title("S Language Studio")

editor = scrolledtext.ScrolledText(root, width=90, height=25, font=("Consolas",
12))
editor.pack(padx=10, pady=10)

btn_frame = tk.Frame(root)
btn_frame.pack()

tk.Button(btnframe, text="تحليل الكود", command=runcode).pack(side=tk.LEFT,
padx=5)
tk.Button(btnframe, text="تشغيل", command=runcode).pack(side=tk.LEFT, padx=5)
tk.Button(btn_frame, text="فتح ملف", command=lambda:
filedialog.askopenfilename()).pack(side=tk.LEFT, padx=5)

root.mainloop()

```

🧠 يمكن تحسين هذا النموذج لاحقًا ليشمل:

- تبويبات للكود والنتائج
- في شجرة رسومية AST عرض الـ
- دمج مستندات الفصل التعليمية في نافذة جانبية

---

📁 IDE دمج الفصول التعليمية داخل

✅ فكرة تبويب "الدروس":

- docs/\*.md يعرض كل فصل من
- يقرأ المحتوى ويعرضه في نافذة جانبية
- المتعلم يقرأ الفصل وينفذ الكود مباشرة

مثال برمجي:

```

python
with open("docs/ch2_basics.md", encoding="utf-8") as f:
    lesson_text.insert(tk.END, f.read())

```

---

📁 الربط مع أدوات المشروع

الوظيفة داخل	IDE الأداة
s_parser.py	AST تحليل الكود إلى
interpreter.py	تشغيل الكود فعليًا
core.slib	استدعاء الدوال الجاهزة
syntax.md	عرض قواعد اللغة المرجعية
ast_samples/	تخزين وتحليل البناء الشجري

> لخلق تجربة كاملة IDE كل هذه الأدوات تُدمج داخل

---

🎯 S Studio رؤية مستقبلية لـ

✅ ميزات متوقعة في الإصدارات القادمة:

- (Syntax Highlighting) التلوين التلقائي للكود
- (Linting + Suggestions) التصحيح الذكي
- (Execute Selection) تشغيل جزئي للكود

- S توصيل عبر الإنترنت لتجارب مجتمع
- HTML / PDF / Markdown تصدير المشاريع إلى

---

تنظيم ملفات الواجهة

...

```
tools/  
└─ ide/  
    ├── s_studio.py  
    ├── assets/  
    │   ├── icon.png  
    │   └── logo.svg  
    ├── templates/  
    │   ├── default.s  
    │   └── welcome.md  
    └──
```

...

---

التصميم، التوثيق، والربط مع المُفسّر - S/S++ الفصل 8: بناء مكتبة قياسية للغة

---

لماذا نحتاج مكتبة قياسية؟

كل لغة برمجة ناجحة تملك مكتبة أساسية توفر دوال جاهزة للتعامل مع النصوص، القوائم، الإدخال، الطباعة، وغيرها.  
تمثل حجر الأساس لبناء التطبيقات، وتجعل اللغة مفيدة S/S++ في لغة core.slib مكتبة منذ أول تجربة.

---

تصميم المكتبة القياسية

stdlib/ المكتبة تُكتب كملف مستقل داخل

```
```bash
stdlib/
└─ core.slib
```
```

وصيغته يمكن أن تكون شبيهة بهيكل اللغة نفسها:

```
```s
func print(text: string);
func input(prompt: string): string;
func len(list: list<any>): int;
func add(a: int, b: int): int;
func split(text: string, sep: string): list<string>;
func type(x: any): string;
```
```

---

توثيق دوال المكتبة

| الدالة                                | التوقيع                             | الوصف العملي |
|---------------------------------------|-------------------------------------|--------------|
| ----- ----- -----                     |                                     |              |
| يطبع النص في المخرجات                 | func print(text: string)            | print        |
| يطلب إدخال من المستخدم كنص            | func input(prompt: string): string  | input        |
| يرجع طول القائمة                      | func len(list: list<any>): int      | len          |
| جمع رقمين صحيحين                      | func add(a: int, b: int): int       | add          |
| تقسيم نص بناءً على فاصل معين          | func split(text, sep): list<string> | split        |
| (int, string...) يرجع نوع المتغير كنص | func type(x: any): string           | type         |

---

ربط المكتبة مع المُفسّر

يُفحص إذا كانت الدالة من المكتبة CallExpression، عند تنفيذ كود من نوع

```
```python
if call["name"] in core_library:
    execute_builtin(call["name"], call["args"])
```
```

:وتحويله إلى سجل دوال متاحة core.slib أو يمكن قراءة

```
```python
```

```
داخل interpreter.py
corefunctions = loadslib("stdlib/core.slib")
```
```

---

📁 مكان توثيق المكتبة في الكتاب

:أنشئ فصل جديد

```
```bash
docs/ch8_library.md
```
```

:ويحتوي

- جدول توثيق الدوال الجاهزة
- استخدام عملي لكل دالة
- examples/ أمثلة مبنية على
- طريقة الربط مع المُفسّر

---

✏️ مثال عملي من المشروع:


```
```s
func main() {
    let name: string = input("؟اسمك");
    print("مرحبًا يا " + name);
}
```
```

.ويقوم بتنفيذهما داخليًا core.slib من print و input يفترض أن المُفسّر يعرف أن


---

✅ خطوات إنشاء الملف:

1. أنشئ الملف stdlib/core.slib
2. اكتب فيه جميع الدوال القياسية بتوقيع واضح
3. كدوال جاهزة interpreter.py اربطها داخل
4. docs/ch8\_library.md وثقها في
5. جرب كل دالة باستخدام أمثلة عملية

# S/S++ الفصل 9: بناء مجتمع حول لغة 

---

## S؟ لماذا نحتاج مجتمعًا للغة 

لغة البرمجة لا تكبر وحدها.

المجتمع هو القوة الدافعة وراء تطوير اللغة، اختبارها، مشاركتها، وتوسيع استخدامها.

كل مساهم، متعلم، أو مراجع يضيف إلى نضج اللغة وموثوقيتها.

---

##  فتح باب المشاركة

### :ملفات يجب توفرها في المستودع

- `CONTRIBUTING.md`: تعليمات للمساهمة
- `CODE\_OF\_CONDUCT.md`: قواعد احترام التفاعل
- لتوثيق المشاكل والاقتراحات: `Issues` صفحة
- لتبادل المعرفة والأفكار: `Discussions` صفحة

### :مساهمات ممكنة

- إضافة دوال قياسية جديدة
- interpreter و parser تحسين
- كتابة فصول أو ترجمات للكتاب
- تطوير أدوات إضافية (محرر، مكتبة...)

---

##  دعم التعليم والتجربة

- توفير أمثلة عملية وسهلة
- دليل "البدء السريع" لكتابة أول كود
- تحديات تعليمية شهريًا (مشروع بسيط بلغتك)
- لعرض الدروس IDE تبويب خاص داخل

---

##  نشر اللغة عالميًا

- (v0.2، v0.3...) إصدارات دورية
- وبلغات متعددة PDF تحويل الكتاب إلى
- Reddit و Dev.to مقالات على منصات مثل
- YouTube فيديوهات تعليمية على

---

##  التواصل بين المطورين والمستخدمين

### :قنوات مقترحة

- GitHub Discussions
- Telegram أو Discord مجتمع
- بريد إلكتروني رسمي لملاحظات المستخدمين
- صفحة "التوثيق المفتوح" لاستقبال تعديلات المجتمع


---

##  الهدف من المجتمع

لغة حيّة، تُطوّر جماعيًا، وتُستخدم في التعليم، المشاريع، والتجارب S/S++ أن تكون لغة > البرمجية حول العالم.




---


##  تذكير للمستخدمين الجدد

- اللغة مفتوحة المصدر
- الكتاب التعليمي جاهز
- أمثلة عملية متوفرة
- دعم موجود والمساهمة مرحّب بها

---

#  مراحل النمو المستقبلية - S/S++ الفصل 10: خارطة تطوير لغة


---

##  هدف هذا الفصل

:بعد إصدار النسخة الرسمية الأولى، تبدأ المرحلة الأهم  
توسيع اللغة، تحسين الأدوات، ودعم ميزات جديدة بشكل منهجي ومنظم.


بحيث تكون كل خطوة مدروسة ، S/S++ هذا الفصل يُحدد خارطة الطريق المستقبلية للغة >  
ولها هدف واضح.

---

##  المرحلة 1: تحسين القواعد الأساسية


- `set` و `tuple` دعم الأنواع المركبة مثل
- تحسين دعم القوائم والخوارزميات عليها
- `match` أو `switch` توسعة نظام الشروط ليشمل
- `format()` إضافة أدوات تنسيق النصوص مثل

---

##  المرحلة 2: دعم البرمجة المتقدمة


- S++ البرمجة الكائنية بالكامل في
- دعم التعددية في الوراثة والتجريد
- توحيد نظام الدوال داخل الكائنات
- (references) دعم المؤشرات المرجعية

---

##  المرحلة 3: بناء مكتبات متخصصة


- للتعامل مع القراءة والكتابة `fs.slib` :مكتبة ملفات
- تشمل دوال الجبر والهندسة `math.slib` :مكتبة رياضية
- للتعامل مع الجمل والكلمات `string.slib` :مكتبة تحليل النصوص
- لتصميم واجهات مرئية `gui.slib` :مكتبة رسومية مستقبلية

---

##  المرحلة 4: تحسين تجربة الاستخدام

- تشمل محرر متقدم S-Studio واجهة
- (Syntax Highlighting) دعم تلوين الكود
- تنفيذ مباشر من داخل المحرر
- تبويب تعليمي يعرض الفصول أثناء الكتابة

---

##  المرحلة 5: تحسين أدوات التنفيذ

- أكثر سرعة وذكاء `interpreter.py` جعل
- تحسين عرض الأخطاء برسائل مفيدة
- دعم تشغيل جزئي (تشغيل جزء من الكود فقط)
- من الكود `exe` توليد ملفات تنفيذية

---

##  المرحلة 6: توسيع الانتشار

- ترجمة الكتاب إلى لغات أخرى (إنجليزي، تركي، فرنسي...)
- إصدار نسخة مستقرة كل 3 شهور
- Dev.to مقالات تعليمية على

- YouTube | إنشاء قناة تعليم على -

---

## 📦 المرحلة 7: بناء نظام حزم ومكتبات خارجية

- (S Package Manager) `spm` تصميم مدير حزم مثل
- GitHub دعم تحميل مكتبات المجتمع عبر
- داخل اللغة `import` إضافة أمر
- بناء مستودع مركزي للمكتبات الجاهزة

---

## 🏠 المرحلة 8: اعتماد اللغة في التعليم

- إصدار نسخة مخصصة للمبتدئين
- تطبيقات تعليمية داخل بيئات المدارس
- للشرح المنهجي PDF كتب وملفات
- S نماذج امتحانات واختبارات في

---

## 📖 الرؤية طويلة المدى

لغة رسمية عالمية، قابلة للتعليم، التوسع، والاستخدام المهني.. وأن S/S++ أن تكون > يُبنى حولها مجتمع نشط يصمم أدوات وأفكار جديدة كل شهر

---

## ✅ تذكير لمطوري اللغة

- `release\_notes.md` كل تحديث يجب أن يكون موثقًا داخل
- كل ميزة جديدة يجب أن تُختبر عمليًا عبر أمثلة
- دعم الاستقرار أهم من زيادة الميزات
- مساهمات المجتمع هي وقود نمو اللغة

---

■ محرّكات، محلّلات، ومترجمات مخصصة – S/S++ الفصل 11: هندسة أدوات متقدمة بلغة

---

🎯 هدف هذا الفصل

.لكتابة مشاريع وتطبيقات بسيطة S حتى الآن استخدمنا لغة:  
لكن قوتها الحقيقية تظهر عندما تُستخدم في بناء أدوات برمجية ذاتية مثل

- محلّلات لغوية (Parser Generators)
- محرّكات قواعد (Rule Engines)
- مترجمات لغات خاصة (DSL Compilers)

لتوليد أدوات تقنية متقدمة قابلة لإعادة S هذا الفصل يُظهر كيف تستخدم لغة > الاستخدام.

---

📦 مشروع 1: توليد محلّل لغوي مصغّر

:الفكرة

.تلقائيًا AST تأخذ تعريفًا لقواعد لغة جديدة وتُنتج ملف S أداة بلغة

:مثال مدخل:

```
```s
let grammar: list<string> = [
  "Expr -> Term + Expr",
  "Expr -> Term",
  "Term -> Number"
];
```

:خطوات التنفيذ

- تحليل الأسطر إلى قواعد
- بناء مصفوفة تحويل
- حفظ الناتج في ملف grammar\_ast.json
- استخدامه لتفسير لغة جديدة

---

🔗 مشروع 2: محرّك قواعد للذكاء الاصطناعي

:الفكرة

.نظام بسيط يُطبق قواعد على حالات معينة لاتخاذ قرارات

:مثال

```
```s++
class Rule {
  let condition: string;
  let action: string;

  func init(c: string, a: string) {
    self.condition = c;
    self.action = a;
  }


  func match(input: string): bool {
    return input.contains(self.condition);
  }
}
```

```
}  
...
```

استخدامات مستقبلية:

- (Expert Systems) أنظمة خبير
- تقييم الردود أو الحالات
- دعم محادثات تعتمد على المنطق

---

 مشروع 3: مترجم لغة مصغرة (Mini DSL)

الفكرة:

اكتب بلغة بسيطة مثل:

```
```txt  
SET x = 5;  
PRINT x + 3;  
```
```


S: لتفسير هذه اللغة وتحويلها إلى أوامر S ثم استخدم

```
```s  
let x: int = 5;  
print(x + 3);  
```
```

فوائد المشروع:

- تدريب على تصميم المترجم
- تعزيز فهم التحليل النصي
- بناء أدوات مخصصة لتعليم البرمجة

---

 ملاحظات هندسية

- دائمًا للمحركات class + func استخدم بنية
- وثق كل مرحلة بتعليقات داخل الكود
- interpreter و AST اربط المشاريع بنظام
- يمكن توليد ملف تنفيذي لكل مشروع لاحقًا

---

 مكان هذه المشاريع في الكتاب

ضع نماذج داخل:

```
```  
examples/tools/  
├─ rule_engine.s++  
├─ mini_parser.s++  
└─ minidslcompiler.s++  
```
```


واربطها بشرح عملي داخل الفصل

---


 خلاصة الفصل

لا تكتب فقط برامج، بل يمكنها أن تصنع أدوات تساعد في بناء لغات وبرمجيات S لغة >

جديدة.  
> هذه هي علامة النص الحقيقي لأي لغة برمجة


# إصدار النسخة الرسمية للتعليم - S/S++ الفصل 12: مشروع مكتمل بلغة 

---

##  ما ستتعلمه في هذا الفصل

:في هذا الفصل، سنطبق كل شيء تعلمناه في الفصول السابقة  
ثم نشرحه خطوة بخطوة، ونبين كيف تُصدر S/S++ سنكتب برنامجًا حقيقيًا باستخدام لغة  
S. مشروع كامل بلغة  
هذا الفصل هو تنويع للرحلة التعليمية التي مررنا بها.

---

##  وصف المشروع

\*\*StudentBook برنامج بسيط يسمى: \*\*دفتر الطالب  
يقوم بـ:

- تخزين بيانات الطالب (اسم، عمر، درجات)
- حساب المعدل النهائي
- طباعة تقرير واضح

---

##  كود المشروع الكامل


```
```s++
class Student {
    let name: string;
    let age: int;
    let grades: list<float>;

    func init(n: string, a: int, g: list<float>) {
        self.name = n;
        self.age = a;
        self.grades = g;
    }

    func average(): float {
        let sum: float = 0;
        for grade in self.grades {
            sum = sum + grade;
        }
        return sum / len(self.grades);
    }


    func report() {
        print("اسم الطالب: " + self.name);
        print("العمر: " + str(self.age));
        print("عدد المواد: " + str(len(self.grades)));
        print("المعدل النهائي: " + str(self.average()));
    }
}

func main() {
    let s: Student = Student("[82.0 ,76.5 ,91.0 ,88.5] ,17 ,\"أحمد\");
    s.report();
}
```

🐧 تشبه السحر للمبرمج المبتدئ؟ S/S++ الفصل 13: لماذا لغة 

---

🎯 لماذا هذا الفصل ممتع؟

المبرمج المبتدئ يحتاج أكثر من القواعد-يحتاج لغة تخاطب روحه البرمجية، تبني ثقته، وتدفعه للاستمرار مع اللغات الأخرى... لكن بأسلوب ممتع، بسيط، S/S++ في هذا الفصل، سنقارن لغة  تحفيزي، وبأمثلة تشع ذكاءً

---

😓 المبرمج المبتدئ يسأل:

- ؟! يقولون إنها قوية C هل أتعلم
- ؟! لأنها سهلة Python أو
- ؟ لأنها مطلوبة في الوظائف؟ Java أم
- ؟ هل هي مناسبة لي؟ S أو... لغة جديدة مثل

صممت لتكون لغة التعلم بمتعة، بدون إرهاق S/S++ الجواب؟ كل لغة لها أسلوبها، بس >

---

🔍 مقارنة لغوية خفيفة ولذيذة

اللغة	هل تحتاج محرك كبير لتشغل؟	هل فيها شروط كثيرة؟	هل الكود يبدو   جميل؟   هل تعلمها ممتع؟
كثير	لا <input checked="" type="checkbox"/> لا <input checked="" type="checkbox"/>	(Compiler) نعم <input checked="" type="checkbox"/>	C   <input checked="" type="checkbox"/> لا <input checked="" type="checkbox"/>
كثير جداً	لا <input checked="" type="checkbox"/> ثقيل <input checked="" type="checkbox"/>	(JVM + IDE) جداً <input checked="" type="checkbox"/>	Java   <input checked="" type="checkbox"/> لا <input checked="" type="checkbox"/>
لا	لا <input checked="" type="checkbox"/> قليلة	مريح <input checked="" type="checkbox"/> ممل بدون مشاريع	Python   <input checked="" type="checkbox"/> لا <input checked="" type="checkbox"/>
فقط ملف واحد!	لا <input checked="" type="checkbox"/> كل شيء بسيط	أنيق <input checked="" type="checkbox"/> إلى حد كبير	S/S++   <input checked="" type="checkbox"/> لا <input checked="" type="checkbox"/>
		+ قابل للقراءة <input checked="" type="checkbox"/> جداً	

---

🔧 ونقارن S... لنكتب كود بلغة

💠 S/S++

```
```s
func main() {
    let name: string = input("ما اسمك؟");
    print("أهلاً يا " + name);
}
```

💠 Java

```
```java
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("ما اسمك؟");
        String name = sc.nextLine();
        System.out.println("أهلاً يا " + name);
    }
}
```



```
}  
}
```

تشعر وكأنك في حفلة أوراق Java أسطر تجيب الابتسامة، بينما في 3 S، هل لاحظت؟ في >  
رسمية 📄 🖨

---

المبتدئ يحتاج لغة مثل 📖

- تكون كل كود فيها واضح
- تعطي نتائج مباشرة
- يمكن تغيير الكود بسهولة
- لا تحتاج حفظ 100 قاعدة قبل كتابة أول برنامج

> .تقدم له ذلك... بصدق، بدون خداع، وبدون رسائل خطأ غامضة S/S++

---

المقارنة من وجهة نظر المبتدئ 🍷

؟ | أمثلة موجودة؟ | هل يشعر المبرمج S/S++ سمة مطلوبة | هل توفرها |  
| بالمتعة؟ |

مدخل واضح للغة	✓ نعم	✓ موجودة بكل فصل	✓ الحماس يبدأ	هل توفرها
سهولة إنشاء مشروع <td>✓ نعم<td>✓ مشاريع بسيطة<td>✓ يمكنه صنع<td>  من أول سطر  </td></td></td></td>	✓ نعم <td>✓ مشاريع بسيطة<td>✓ يمكنه صنع<td>  من أول سطر  </td></td></td>	✓ مشاريع بسيطة <td>✓ يمكنه صنع<td>  من أول سطر  </td></td>	✓ يمكنه صنع <td>  من أول سطر  </td>	من أول سطر
كائنات برمجية حقيقية <td>✓ مدعومة<td>✓ أمثلة الطالب والصندوق<td>✓ لعبة صغيرة بنفسه<td>  يشعر  </td></td></td></td>	✓ مدعومة <td>✓ أمثلة الطالب والصندوق<td>✓ لعبة صغيرة بنفسه<td>  يشعر  </td></td></td>	✓ أمثلة الطالب والصندوق <td>✓ لعبة صغيرة بنفسه<td>  يشعر  </td></td>	✓ لعبة صغيرة بنفسه <td>  يشعر  </td>	يشعر
			بأنه بنى شيئًا محترمًا	

---

❗ "مثال يجعل المبرمج يقول "أنا عبقرى"

```
```s++  
class Game {  
    let title: string;  
    func init(t: string) {  
        self.title = t;  
    }  
    func start() {  
        print("🎮 تبدأ اللعبة " + self.title);  
    }  
}  
  
func main() {  
    let g: Game = Game("مغامرات النينجا");  
    g.start();  
}  
```
```

> لا يحتاج أن يفهم الوراثة ولا الذاكرة المؤقتة-هو فقط كتب لعبة، وشغلها... وهو فخور بها.

---

🎯 لماذا هذا الفصل مهم؟

:لأنه يخبر كل مبرمج مبتدئ  
> ...أنت لست بحاجة أن تكون بطل المسابقات لتكتب كود جميل  
> ".فقط استخدم لغة تتحدث إليك، بلغة قلبك قبل عقلك"

---

### ✓ خلاصة الفصل 13:

التأثير | S السبب الذي يجعل المبرمج المبتدئ يحب لغة

-----|-----

-----|

الكود قصير وواضح 😊 | يراه، يفهمه، لا يهرب منه

لا يحتاج مكتبات معقدة | يركز على الفكرة لا على التهيئة

بدون شروحات مطولة print() المكتبة القياسية شبيهة بلغات مشهورة | يفهم

كل مشروع فيه هدف حقيقي | يشعر أنه يُنتج، لا ينسخ فقط

---

### 🔥 رسالة ختامية

- > ...عزيزي المبرمج
- > ،ليست فقط لتكتب فيها كود S/S++ لغة
- > ! بل لتبدأ رحلتك، تشعر بالإنجاز، وتستمتع بتجربة بناء الأفكار
- > 🌟💡📚 وإذا ضحكت وأنت تتعلم، فأنت في الطريق الصحيح

## ■ S/S++ الفصل 14: ألعاب صغيرة، تطبيقات ذكية، وحماية بسيطة بلغة

---

### 🎯 هدف الفصل

كيف تصمم كائنات، كيف تنفذ مشاريع، لكن... ما رأيك S، تعلمت الآن كيف تكتب كود بلغة أن تنتقل إلى تطبيقات خفيفة، ممتعة، وتعزز معرفتك بالأمان البرمجي؟ ✨! هنا نبدأ ببناء ألعاب بسيطة، أدوات حاسبة، وحتى أنظمة تحقق وكلمات مرور

---

### 🎮 1. لعبة التخمين - Guess the Number

#### ✅ الكود:

```
`s
func main() {
    let answer: int = rand();
    print("🔒 خمن رقمًا بين 0 و100");
    while true {
        let guess: int = int(input("رقمك؟"));
        if guess == answer {
            print("✅ صحيح");
            break;
        } else {
            print("❌ خطأ، جرب مجددًا");
        }
    }
}
```

#### 🎯 التعلم:

- لإنشاء رقم سري rand() استخدمت
- للتكرار while استخدمت حلقة
- أدخلت من المستخدم رقمًا
- أضفت منطق حماية من الإدخال الخاطئ

---

### 🧮 2. تطبيق آلة حاسبة ذكية

```
`s
func add(a: int, b: int): int {
    return a + b;
}
func main() {
    let x: int = int(input("رقم أول: "));
    let y: int = int(input("رقم ثاني: "));
    let result: int = add(x, y);
    print("✅ الناتج: " + str(result));
}
```

> يمكن توسعة المشروع ليشمل عمليات الطرح، الضرب، القسمة... بل وحتى معادلات

---

### 🔑 3. أداة التحقق من كلمة المرور

```
`s
func main() {
    let stored: string = "s2pass@2025";
```

```

let input_pass: string = input("🔑 كلمة المرور : ");
if input_pass == stored {
    print("✅ الوصول مسموح");
} else {
    print("❌ كلمة مرور خاطئة");
}
}
`

```

🔒 لماذا هذا مفيد؟

- تعريف أمان بسيط
- حماية من الوصول غير المصرح
- يمكن إضافة محاولات محدودة، أو تشفير لاحقًا

---

⚠️ ملاحظات الحماية للمبتدئين

- لا تخزن كلمة المرور كنص واضح في الملفات
- استخدم دوال تحقق أو تشفير لاحقًا
- راقب الإدخال دائمًا ولا تنفذ مباشرة بناءً على إدخال المستخدم

---

🧠 S/S++ فكرة إضافية: لعبة "الذاكرة" بلغة

> . تطابق بطاقتين تحتويان رموزًا، عليك تذكر موقعها

```

`s
func main() {
    let data: list<string> = ["🐱", "🐶", "🐱", "🐶"];
    let shuffle: list<string> = reverse(data);
    print("🧠 اختبر ذاكرتك، أين الرموز؟");
    // أكمل اللعبة حسب إدخال المستخدم
}
`

```

---

📦 كيف يساهم هذا الفصل في الأمن البرمجي؟

| الأداة/اللعبة  | مبدأ الحماية المتعلم           |
|----------------|--------------------------------|
| لعبة التخمين   | نظام تحقق متكرر بدون كشف أسرار |
| كلمة المرور    | مفهوم المصادقة                 |
| الحاسبة الذكية | فحص نوع البيانات               |
| لعبة الذاكرة   | إخفاء المعلومات مؤقتًا         |

---

✅ ماذا أستفيد من هذا الفصل؟

- S تفكير تطبيقي باستخدام لغة
- تعلم أساسيات الحماية داخل كود مبسط
- صناعة تطبيقات تفاعلية للمستخدم
- بناء برامج بها منطق، وليس فقط طباعة

---

🎯 مهمة الفصل (للمبرمج الحقيقي)

> يطلب من المستخدم تسجيل اسم مستخدم وكلمة مرور، ثم يتحقق S اكتب تطبيقًا بلغة > منها.

- > .أضف نظام تأمين مؤقت بعد ثلاث محاولات خاطئة
  - > .أضف طباعة رمز عشوائي بعد كل تسجيل دخول ناجح
-

التشفير، حماية البيانات، وأمان - S/S++ الفصل 15: بناء مشاريع قوية باستخدام السيرفرات

---

## 🎯 هدف الفصل

في هذا الفصل، ننتقل من التطبيقات الخفيفة إلى أدوات حقيقية تمس عالم البرمجة الاحترافية.

سنتعلم:

- S/S++ كيفية بناء تطبيقات كبيرة بلغة
- كيفية تصميم أدوات تشفير أساسية
- كيف نحمي البيانات أثناء المعالجة
- آليات مراقبة دخول وتفاعل المستخدم
- مبادئ الحماية على مستوى الخادم (السيرفر)

---

## 🏠 S هيكل مشروع كبير بلغة 1.

مثال مشروع: نظام تسجيل دخول آمن + تشفير + سجل زيارات

```
```bash
secure_app/
├── main.SPP
├── crypto.SPP
├── users.s
├── log.s
└── stdlib/core.slib
```
```

.كل ملف مسؤول عن جزء من النظام: واجهة، تشفير، بيانات، سجل

---

## 🔒 بناء وحدة تشفير بسيطة 2.

```
```s
func hash(password: string): string {
    // تشفير بسيط باستخدام انقلاب النص
    return reverse(password) + "_s2hash";
}
```
```

داخل AES أو SHA لاحقًا يمكن بناء مكتبة تشفير متقدمة باستخدام خوارزميات مثل > crypto.SPP

---

## 🔑 حماية بيانات المستخدم 3.

```
```s++
class User {
    let username: string;
    let password: string;

    func init(u: string, p: string) {
        self.username = u;
        self.password = hash(p);
    }

    func check(login: string): bool {
        return login == self.password;
    }
}
```

```
}
}...
```

> تخزين كلمات المرور مشفرة، ومنع المقارنة المباشرة بالنص الحقيقي

---

#### 📁 4. سجل زيارات المستخدمين

```
```s
func log_visit(user: string) {
    let timestamp: string = str(rand()); // لاحقًا استخدم وقت حقيقي
    print("📄 دخول: " + user + " في " + timestamp);
}
```
```

> يمكن مستقبلاً حفظ هذه البيانات في ملف أو إرسالها لسيرفر خارجي

---

#### 🛡️ 5. حماية السيرفر من الزوار غير المصرّح لهم

خطوات:

1. يحتوي قائمة المستخدمين المصرّح لهم users.s إنشاء ملف
2. تنفيذ فحص عند كل دخول جديد
3. إضافة تأخير زمني عند فشل الدخول
4. إغلاق النظام بعد عدد محاولات خاطئة متتالية

---

#### 💡 مشروع: بوابة مشفرة للرسائل 6.

```
```s++
class Message {
    let content: string;

    func init(c: string) {
        self.content = c;
    }

    func encrypt(): string {
        return reverse(self.content) + "@lock";
    }

    func decrypt(enc: string): string {
        return reverse(enc.replace("@lock", ""));
    }
}
```
```

> هذا المشروع يعلم المبرمج كيف تُخزّن البيانات بطريقة آمنة، وتُفكّر كأداة أمن معلومات مصغرة.

---

#### 🛡️ 7. ملاحظات حماية مهمة

| تقنية الحماية |                                       | لماذا تُستخدم |
|---------------|---------------------------------------|---------------|
| التشفير       | عدم كشف البيانات الحساسة              |               |
| التجزئة       | حماية كلمات المرور بدون إمكانية العكس |               |
| سجل الدخول    | مراقبة التفاعل والاختراقات المحتملة   |               |

| (brute-force) تأخير المحاولة | منع الهجمات المتكررة |

---

✅ ماذا بعد هذا الفصل؟

المبرمج الذي أنهى هذا الفصل يمكنه الآن تصميم أدوات حقيقية تحمي المستخدمين، >  
> وتفكر بنمط مؤسسي.  
> أصبح لديك القدرة على صناعة أدوات تشفير، مصادقة، وحماية تطبيقاتك من الداخل >

---

🎯 مهمة الفصل:

: يحوي الآتي S/S++ اكتب مشروعًا بلغة

- صفحة دخول
- تخزين كلمات المرور المشفرة
- سجل زمني لكل دخول ناجح أو فاشل
- إغلاق الحساب بعد ثلاث محاولات خاطئة
- تشفير الرسائل بين المستخدم والخادم

---



---

## 🎯 هدف الفصل

بعد أن تعلمت كيف تبني تطبيقات مشفرة وتحمي البيانات، حان الوقت للدخول في عالم الأمن السيبراني:  
كيف تُفكّر كمُدافع؟ كيف تُصمم أدوات أمن؟ كيف تُحلّل التطبيقات وتكشف الثغرات؟ >  
في هذا الفصل، تبدأ رحلتك نحو الفهم العميق للهندسة العكسية، تحليل السلوك البرمجي، وتصميم أدوات مخصصة بالأمان والهجوم.

---

## 🔍 ما هو الأمن السيبراني؟

هو علم حماية الأنظمة، الشبكات، والتطبيقات من التهديدات والهجمات:  
يتضمن:

- تحليل السلوك البرمجي
- كشف الثغرات ونقاط الضعف
- الرد على الهجمات واستعادة البيانات
- تصميم أدوات دفاع واختبار

---

## 🧠 ما هي الهندسة العكسية؟

هي عملية تحليل برنامج أو تطبيق لتفكيك مكوناته وفهم سلوكه الداخلي بدون الوصول إلى كود المصدر.

وكشف سلوك تنفيذ معين، AST، يمكنك كتابة أدوات لفحص الكود، تحليل S/S++ في لغة >

---

## 📁 S لكود AST مشروع: تحليل

ملف: reverse\_tool.SPP

```
```s++
class Analyzer {
    let code: string;

    func init(c: string) {
        self.code = c;
    }

    func tokenize(): list<string> {
        // تفكيك الكود إلى رموز أولية
        return split(self.code, " ");
    }

    func detect_calls(): list<string> {
        let tokens: list<string> = self.tokenize();
        let calls: list<string> = [];
        for t in tokens {
            if t == "print" or t == "input" {
                push(calls, t);
            }
        }
        return calls;
    }

    func report() {
```

```

        let result: list<string> = self.detect_calls();
        print("تم العثور على الدوال التالية");
        for r in result {
            print("- " + r);
        }
    }
}

func main() {
    let source: string = "let x = input(); print(x);";
    let a: Analyzer = Analyzer(source);
    a.report();
}

```

. هذا مثال بسيط لتحليل الشيفرة بحثًا عن دوال معينة-نواة لأداة فحص سلوك >

---

S أدوات الأمان التي يمكنك تصميمها في 🛡️

الأداة	الوصف
eval, exec	كشف الدوال الحساسة   تحليل الكود بحثًا عن دوال مثل
input()	لمعرفة من يُدخل البيانات   تتبع المدخلات   مراقبة
كشف الحلقات الزائدة	اكتشاف الحلقات الزائدة   كشف الحلقات غير المحدودة
مصفاة الكلمات	حذف أو تعديل كلمات غير مسموح بها
مراقب التعديل	مقارنة بين نسختين من ملف لكشف الاختراق

---

AST أداة الهندسة العكسية - فك □

ويُعيد بناء الكود خطوة بخطوة بصيغة قابلة main\_ast.json يمكنك بناء برنامج يقرأ ملف للفرءة.

.الهدف: فهم بنية التنفيذ من دون الدخول إلى الشيفرة المصدر >

---

مسؤوليات الأخلاقيات ⚠️

:الأمن السيبراني لا يعني الاختراق فقط، بل يعني

- الحماية قبل الهجوم
- الاختبار الشرعي، وليس العبث غير القانوني
- تطوير أدوات تُفيد الآخرين، لا تُؤذيهم
- بناء نظام يرد على الهجمات، وليس ينفذها

---

S مثال أدوات "قوية" بلغة 🗨️

الأداة	الاستخدام
logger.SPP	تسجيل كل طلب إدخال للمستخدم
obfuscator.SPP	إعادة كتابة كود ليصعب قراءته
integrity.SPP	التحقق من الملف قبل التنفيذ
patcher.SPP	AST تعديل كود عن طريق

---

التدريب العملي 🛠️

> فُكِّر في سيناريو :  
> هناك برنامج مشكوك فيه ، يحتوي دالة لا يجب أن تعمل إذا لم يكن المستخدم مصرحاً  
> مستخدمة دون شرط unlock() وتكتشف إن كانت main.s، تقوم بفحص ملف S اكتب أداة  
تحقق .





---

#### ✓ خلاصة الفصل

: قد تبدو بسيطة ، لكن تستطيع استخدامها لبناء أدوات S/S++ لغة

- تحليل سلوك البرامج
- اكتشاف الحالات المشبوهة
- (Obfuscation) تعمية الكود
- هندسة عكسية على مستوى التعليم
- تدريب على الدفاع الرقمي بأسلوب عملي

---

الفصل 17: استخدام أدوات الأمن السيبراني بالشكل الصحيح - المسؤولية، الأخلاق،  والتطبيق العملي   

---

### هدف الفصل

:بعد أن تعلّمت كيف تصنع أدوات تحليل وهندسة عكسية، حان الوقت لتعلّم الشيء الأهم.  
كيف تستخدم هذه الأدوات بشكل أخلاقي، مسؤول، وفَعّال داخل بيئة الأمن السيبراني >

هذا الفصل ليس تقنيًا فقط، بل أخلاقي أيضًا-لأنه يُعلّمك أن تكون مدافعًا ذكيًا، لا مهاجمًا عشوائيًا.

---

### من يستخدم أدوات الأمن؟

- المبرمج الذي يبني تطبيقًا آمنًا
- محلل أمن يختبر برامج قبل إطلاقها
- الباحث الأمني الذي يكشف نقاط الضعف ثم يبلغ عنها
- المعلم الذي يدرّب طلابه على حماية الأنظمة بكفاءة

يمكن استخدامها داخل هذه السيناريوهات... لكن S/S++ الأدوات التي طوّرتها بلغة >  
بشروط ⚠

---

### المبادئ الأخلاقية في استخدام أدوات الأمن

المبدأ	الوصف
إلا تفحص تطبيقًا أو نظامًا بدون تصريح واضح	إلا تفحص تطبيقًا أو نظامًا بدون تصريح واضح
هدف الحماية	هدف الحماية
البلاغ الأمني	عند اكتشاف ثغرة، أبلغ المسؤول عنها رسميًا
إلا تشارك كود أداة هجومية دون توثيق واضح أو قيود	عدم النشر الخطير

---

### كيف تستخدم أدواتك التي كتبتها؟

:مثلًا SPP.logger أداة

- استخدمها داخل تطبيقك الخاص
- لتسجيل مدخلات المستخدم في حالة التجاوز أو المحاولات الفاشلة
- لا تُرسل هذه البيانات إلى الخارج بدون إذن المستخدم

:SPP.reverse\_tool أداة

- داخل فصلك أو شركتك S استخدمها لتحليل ملفات
- لا تستخدمها على مشاريع خارجية لم تطوّرها أنت

---

### خطوات الاستخدام المسؤول لأي أداة أمنية

1. وضح هدف الأداة داخل الوثائق  
"أكتب في أعلى الملف: "هذه الأداة تستخدم لتحليل كود شخصي لأغراض التدريب"

2. أضف تحذير واضح عند التنفيذ  
:أضف main() داخل

```s

print("⚠ هذا الكود يُستخدم لأغراض أمنية شرعية فقط");

3. لا تربط الأداة بشبكة خارجية بدون تصريح.

4. احفظ بيانات المستخدم المشفرة فقط.

---

مثال استخدام عملي داخل شركة ☞

.لديك تطبيق داخلي في الشركة، وتريد اختبار الأمان فيه قبل النشر >  
.وتحليل المتغيرات والأنشطة المشبوهة AST لفحص S/S++ تستخدم أداة >  
> بعد كل فحص، يتم إرسال تقرير داخلي إلى قائد الفريق ويُراجع النتائج-بشكل رسمي >  
.ومراقب.

---

هل يمكن لتلميذ أن يستخدم هذه الأدوات؟ 🧠

:نعم، لكن بشروط

- داخل بيئة مغلقة
- مع كود كتبته بنفسه
- تحت إشراف معلم أو داخل اختبار محدد

يمكن استخدامها لتعليم الأمن السيبراني من المرحلة S/S++ وهنا تكمن روعة لغة >  
الأولى.

---

الأمنية؟ S/S++ كيف تُعلم غيرك استخدام أدوات 📖

- "أنشئ فصلاً داخل الكتاب التعليمي بعنوان "اختبار التطبيق قبل النشر"
- أضيف أمثلة واقعية: سجل دخول، تحقق من الإدخال، تحليل الكود
- لعرض الكود والتحليل S-Studio استخدم واجهة محرر مثل
- "قدم تحديثات تعليمية للمستخدم: "اكتشف دالة مشبوهة"

---

تحذيرات رسمية 🚫

- أدوات الأمن ليست لعبة: لا تُستخدم خارج الإطار الشرعي
- كل اختبار يجب أن يتم في بيئة تجريبية
- لا تدمج أدوات هجومية مع أدوات تطبيق دون فصل واضح
- لا تنشر أدوات "اختراق" بدون حماية أو توثيق دقيق

---

🟢 خلاصة الفصل 17:

النقطة الأساسية | المعنى العملي |

-----|-----

-----|

أدوات الأمن يجب استخدامها بذكاء | لا تكسر، بل اختبر |

الأخلاق قبل التقنية | الإذن ثم التحليل، لا العكس |

تصلح للتعليم الأمني | لأنها واضحة، وقابلة للتوثيق، ومنضبطة S/S++ لغة |

المبرمج مسؤول عن أفعاله | والكود الذي يكتبه يمكنه أن يحمي أو يُخترق، حسب |  
نواياه |

---

🎯 المهمة الختامية لهذا الفصل:

:تُظهر في تقريرها S اكتب أداة تحليل بلغة

- input() عدد دوال
- إن وجدت eval() عدد دوال
- (مثل while true) عدد الحلقات غير المحددة
- وتنتهي بطباعة:  
"تم تنفيذ التحليل، استخدم النتائج بشكل قانوني فقط ⚠"

---

🧑🏫📦🚀 متكاملة - اللغة، الأدوات، المفسر، والمحرر S الفصل 18: بناء منصة

---

## 🎯 الهدف من الفصل

كل ما بنيته خلال الفصول السابقة كان أجزاء منفصلة: ملفات اللغة، مكتبة قياسية،...مفسر، أدوات، محرر  
لكن الآن، حان وقت دمج كل هذه المكونات في منصة واحدة تُدعى:

> 💡 S-Platform

> S/S++ نظام متكامل لتعليم، تجربة، وتحليل كود بلغة

---

## 📦 مكونات المنصة

| الوصف            | العنصر                               |
|------------------|--------------------------------------|
| ملفات            | كمصدر برمجي SPP، S، اللغة            |
| سكرت             | لتشغيل الكود interpreter.py المفسر   |
| المكتبة القياسية | لدوال جاهزة stdlib/core.slib         |
| الأمثلة          | لتعليم المستخدمين examples/          |
| الأدوات الأمنية  | لتحليل الكود وحمايته tools/security/ |
| المحرر           | كمحرر رسومي tools/ide/s_studio.py    |
| صفحات التوثيق    | للفصول والشرح الرسمي docs/           |
| ملفات النظام     | إلخ README.md، LICENSE               |

---

## 📁 هيكل المشروع المقترح

...

```
S-Platform/
├── stdlib/
│   └── core.slib
├── parser/
│   ├── interpreter.py
│   └── s_parser.py
├── examples/
│   ├── basics/
│   ├── oop/
│   └── projects/
├── tools/
│   ├── ide/
│   │   └── s_studio.py
│   └── security/
│       └── reverse_tool.SPP
├── docs/
│   ├── ch1_intro.md
│   ├── ...
│   └── ch18_platform.md
├── LICENSE
└── README.md
```

...

> كل جزء من المنصة يخدم هدفًا محددًا: تشغيل، تعليم، فحص، تطوير، حماية

---

## 🧠 ما تقدمه S-Platform؟

- محرر رسومي جاهز للتجربة
- مفسر يدعم اللغة ويشغل ملفاتك بسهولة

- مكتبة قياسية تدعم البرمجة الفعلية
- أمثلة تعليمية تغطي معظم المفاهيم
- أدوات تحليل أمنية وهندسة عكسية
- فصول موثقة لكل مستوى تعليمي

---

🔧 كيف تُجهّز المنصة للتشغيل؟

1. فكّر كمستخدم مبتدئ

- افتح s\_studio.py
- يكتب كودًا داخل المحرّر
- يضغط زر تشغيل
- يشاهد النتيجة مباشرة

2. فكّر كمطوّر محترف

- لتجربة تنفيذ جديد interpreter.py يعدّل
- يضيف دالة داخل core.slib
- يكتب مشروع داخل examples/projects/
- يربط أداة أمان داخل tools/security/

---

📄 docs/ch18\_platform.md : الفصل الخاص

.الرسمي S هذا الفصل (أنت تقرأه الآن)، سيكون جزءًا أساسيًا داخل كتاب :يمكن أن تبدأ برسالة قوية

> .اللغة الحقيقية ليست مجرد قواعد... بل نظام كامل"  
> ".الآن لم تعد فكرة، بل منصة تعليمية قابلة للتطوير عالميًا S و"

---

✅ مميزات المنصة التعليمية

| العنصر           | لماذا مهم؟                        |
|------------------|-----------------------------------|
| التجميع الكامل   | كل شيء في مكان واحد               |
| قابلية التوسعة   | أي مبرمج يمكنه إضافة دالة أو أداة |
| واجهة رسومية     | تجعل تجربة التعليم ممتعة وواقعية  |
| أدوات الأمان     | تضيف مسؤولية ووعي لكل مشروع       |
| التوثيق التفصيلي | يساعد المعلمين والطلاب معًا       |

---

🎯 المهمة الختامية

:وانقل إليه S-Platform، أنشئ مجلدًا جديدًا باسم

- ملفات اللغة
- المفسّر والأدوات
- الأمثلة والمحرّر
- ملفات التوثيق والمكتبة
- SLanguageBook\_v0.1.pdf :صفحة غلاف للكتاب باسم

:ثم اكتب

".المنصة التي تبدأ من التعليم، وتصل بك إلى الاحتراف S-Platform... مرحبا بك في">

---