

تطبيق لآلة حاسبة بسيطة بلغة Python

الطالب : لؤي بهاء الدين زلي

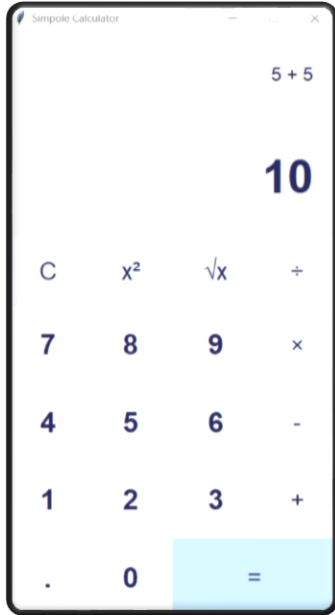


التعريف بالتطبيق



- ✓ واجهة مستخدم بسيطة لآلة حاسبة تم إنشاؤها باستخدام مكتبة **tkinter** في لغة البرمجة **Python** الآلة الحاسبة تحتوي على منطقة عرض لعرض المدخلات الذي يتم إدخالها من قبل المستخدم والنتيجة الإجمالية.
- ✓ كما تحتوي على أزرار للأرقام من 0 إلى 9 والعمليات الحسابية المختلفة.

مميزات التطبيق



- ✓ يمكن للمستخدم إدخال الأرقام والعمليات الحسابية عن طريق النقر على الأزرار من واجهة التطبيق أو لوحة المفاتيح.
- ✓ يتم عرض الأرقام المدخلة والمجموع في الجزء السفلي من منطقة العرض.
- ✓ يتم عرض العمليات الحسابية في الجزء العلوي من منطقة العرض.
- ✓ يمكن للمستخدم مسح المدخلات الحالية والإجمالي عن طريق النقر على زر (C).
- ✓ يمكن للمستخدم حساب تربيع الرقم عن طريق النقر على زر (X^2).
- ✓ يمكن للمستخدم حساب الجذر التربيعي للرقم عن طريق النقر على زر ($\sqrt{x}$).
- ✓ يمكن للمستخدم حساب نتيجة التعبير الحالي عن طريق النقر على زر (=).
- ✓ التطبيق لا يقبل القسمة على العدد (0) حيث تظهر النتيجة (Error).

شرح كود التطبيق

```
import tkinter as tk

LARGE_FONT_STYLE = ("Arial", 40, "bold")
SMALL_FONT_STYLE = ("Arial", 16)
DIGITS_FONT_STYLE = ("Arial", 24, "bold")
DEFAULT_FONT_STYLE = ("Arial", 20)

OFF_WHITE = "#F8FAFF"
WHITE = "#FFFFFF"
LIGHT_BLUE = "#CCEDFF"
LIGHT_GRAY = "#F5F5F5"
LABEL_COLOR = "#25265E"
```

✓ تم استيراد مكتبة Tkinter
✓ تم تعريف بعض الأنماط والألوان والثوابت التي سيتم استخدامها في واجهة التطبيق.

شرح كود التطبيق

```
class Calculator:  
    def __init__(self):  
        self.window = tk.Tk()  
        self.window.geometry("375x667")  
        self.window.resizable(0, 0)  
        self.window.title("Simple Calculator")
```

- ✓ تم تعريف class للآلة الحاسبة.
- ✓ تم تعريف دالة البناء لتجهيز الآلة الحاسبة للتشغيل.
- ✓ تم إنشاء نافذة التطبيق.
- ✓ تم تحديد مقاسات التطبيق لجعله غير قابل للتغيير.
- ✓ تم تسمية عنوان التطبيق (Simple Calculator)

شرح كود التطبيق

```
self.total_expression = ""  
self.current_expression = ""  
self.display_frame = self.create_display_frame()  
  
self.total_label, self.label = self.create_display_labels()
```

- ✓ تم تهيئة متغيرات لتخزين التعابير الكاملة والتعبيرات الحالية.
- ✓ تم إنشاء إطار عرض التعبيرات الحالية والكاملة.
- ✓ تم إنشاء علامات لعرض التعابير الكاملة والتعبيرات الحالية.

شرح كود التطبيق

✓ تم تهيئة الأرقام وموقعها على واجهة التطبيق.

```
self.digits = {  
    7: (1, 1), 8: (1, 2), 9: (1, 3),  
    4: (2, 1), 5: (2, 2), 6: (2, 3),  
    1: (3, 1), 2: (3, 2), 3: (3, 3),  
    0: (4, 2), '.': (4, 1)  
}
```

شرح كود التطبيق

```
self.operations = {"/": "\u00F7", "*": "\u00D7", "-": "-", "+": "+"}  
self.buttons_frame = self.create_buttons_frame()
```

- ✓ تم تهيئة العمليات الحسابية ورموزها.
- ✓ تم إنشاء إطار لوضع الأزرار.

شرح كود التطبيق

```
self.buttons_frame.rowconfigure(0, weight=1)
for x in range(1, 5):
    self.buttons_frame.rowconfigure(x, weight=1)
    self.buttons_frame.columnconfigure(x, weight=1)
self.create_digit_buttons()
self.create_operator_buttons()
self.create_special_buttons()
self.bind_keys()
```

```
self.reset_button()
```

```
self.clear_button()
```

- ✓ تم تكوين صفوف الأزرار للتمدد بنسبة معينة.
- ✓ تم إنشاء الأزرار الخاصة بالأرقام.
- ✓ تم إنشاء الأزرار الخاصة بالعمليات الحسابية.
- ✓ تم إنشاء الأزرار الخاصة بالوظائف الخاصة مثل المساواة والمسح وغيرها.
- ✓ تم ربط مفاتيح لوحة المفاتيح بالأزرار المقابلة.

شرح كود التطبيق

✓ تم تعريف دالة ربط المفاتيح بالأحداث والعمليات.

```
def bind_keys(self):
    self.window.bind("<Return>", lambda event: self.evaluate())
    for key in self.digits:
        self.window.bind(str(key), lambda event, digit=key: self.add_to_expression(digit))

    for key in self.operations:
        self.window.bind(key, lambda event, operator=key: self.append_operator(operator))
```

شرح كود التطبيق

```
def create_special_buttons(self):  
    self.create_clear_button()  
    self.create_equals_button()  
    self.create_square_button()  
    self.create_sqrt_button()
```

- ✓ تم إنشاء أزرار خاصة مثل مسح الشاشة وربطها بدوالها:
- ❖ مسح التعبير الحالي والتعبير الكلي.
- ❖ حساب التربيع للقيمة الحالية.
- ❖ حساب الجذر التربيعي للقيمة الحالية.
- ❖ حساب النتيجة النهائية للتعبير الكلي وتحديث العرض.

شرح كود التطبيق

```
def create_display_labels(self):
    total_label = tk.Label(self.display_frame, text=self.total_expression, anchor=tk.E, bg=LIGHT_GRAY,
                           fg=LABEL_COLOR, padx=24, font=SMALL_FONT_STYLE)
    total_label.pack(expand=True, fill='both')

    label = tk.Label(self.display_frame, text=self.current_expression, anchor=tk.E, bg=LIGHT_GRAY,
                    fg=LABEL_COLOR, padx=24, font=LARGE_FONT_STYLE)
    label.pack(expand=True, fill='both')

    return total_label, label

def create_display_frame(self):
    frame = tk.Frame(self.window, height=221, bg=LIGHT_GRAY)
    frame.pack(expand=True, fill="both")
    return frame

def add_to_expression(self, value):
    self.current_expression += str(value)
    self.update_label()

def create_digit_buttons(self):
    for digit, grid_value in self.digits.items():
        button = tk.Button(self.buttons_frame, text=str(digit), bg=WHITE, fg=LABEL_COLOR, font=DIGITS_FONT_STYLE,
                           borderwidth=0, command=lambda x=digit: self.add_to_expression(x))
        button.grid(row=grid_value[0], column=grid_value[1], sticky=tk.NSEW)

def append_operator(self, operator):
    self.current_expression += operator
    self.total_expression += self.current_expression
    self.current_expression = ""
    self.update_total_label()
    self.update_label()
```

✓ ربط الازرار بدوالها واعداداتها.

شرح كود التطبيق

```
def run(self):  
    self.window.mainloop()  
  
if __name__ == "__main__":  
    Simpel_Culculator = Calculator()  
    Simpel_Culculator.run()
```

- ✓ تم تعريف الدالة (run) ليتم استدعاء واجهة البرنامج.
- ✓ تم استخدام (if __name__ == "__main__") هذا الشرط عندما يتم تشغيل البرنامج كبرنامج رئيسي وليس كمكتبة تم استيرادها.
- ✓ تم إنشاء متغير واسناده (class Calculator).
- ✓ تم ربط المتغير بدالة (run) وذلك لتشغيل التطبيق



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY

تم بحمد الله
وتوفيقه

