

# Programmation

SIE/GCG, Cours 13

27 mai 2024

Jean-Philippe Pellet

```

class ProgramView(Canvas):
    def __init__(self, parent, title) -> None:
        Canvas.__init__(self, parent, height=2 * TOP_MARGIN + 4 * LINE_HEIGHT, highlightthickness=0, title=title)

    def redraw(
        self,
        program: Program,
        current_subprogram: List[Instruction],
        current_instruction_index: int,
    ) -> None:
        height = self.winfo_height()
        width = self.winfo_width()

        self.delete(ALL)
        self.create_rectangle(0, 0, width, height, fill=window_background_color, width=0)

        # boucle pour les 4 sous-programmes P1 à P4
        for i, subprogram in enumerate(
            [program.P1, program.P2, program.P3, program.P4]
        ):
            # dessin du titre
            instruction_center_y = TOP_MARGIN + 1 * LINE_HEIGHT + LINE_HEIGHT // 2
            self.create_text(LEFT_MARGIN // 3, instruction_center_y, text=f"P{i + 1}")

            # dessin de chaque instruction
            for j, instr in enumerate(subprogram):
                instruction_center_x = (
                    LEFT_MARGIN
                    + j * (INSTRUCTION_BOX_SPACING + INSTRUCTION_BOX_WIDTH)
                    + INSTRUCTION_BOX_WIDTH // 2
                )
                instruction_center_y = TOP_MARGIN + 1 * LINE_HEIGHT + LINE_HEIGHT // 2
                self.create_text(instruction_center_x, instruction_center_y, text=instr)

```

# Previously, on ICC Programmation...

---

- **Types** de base en Python: `int`, `float`, `str`, `bool`
- **Méthodes, fonctions et slicing** pour calculer des valeurs dérivées
- **Conditions** pour exécuter du code selon la valeur d'une expression booléenne
- **Boucles** pour exécuter du code plusieurs fois:
- **Déclaration de fonctions** avec type de retour et paramètres:
- Utilisation de **listes, sets** et **dictionnaires**
- Déclaration de **classes** et de méthodes
- Création, chargement, manipulation et sauvegarde d'**images**
- **Programmation dynamique** pour trouver des seams
- Fonctions comme **valeurs, paramètres**; fonctions **d'ordre supérieur**
- Lecture et écriture de **fichiers**
- Manipulations de **chaînes de caractères**
- Concept des **threads**: exécution concurrente/parallèle, locks, deadlocks

# Miniprojet

---

- **Rappel:** **Inscription aux groupes** en ligne, il manque 2 inscriptions (141/143)
  - Délai d'inscription: **(avant) aujourd'hui**
- **Délai de rendu: 31 mai 2024 à 23h59** (reçu 48/86)
  - En ligne en sur la dernière semaine sur Moodle
- Un seul fichier à rendre: **miniproject.py**
- **Un seul rendu** par groupe (membre 1 du groupe)
- Evaluation: uniquement de vos **fonctions à compléter**
  - Pas du “main”
- **Pas de code aligné tout à gauche**

# Examen final

---

- **24 juin 2024, 15h**, salles communiquées plus tard
- $\frac{1}{3}$  programmation
  - 8 QCM
  - 1 question ouverte, programmation sur papier
- $\frac{2}{3}$  théorie
  - Selon indications d'Olivier Lévêque

# Révisions

# Thèmes

---

- 8 × classes et dictionnaires
- 6 × fonctions lambda
- 1 × arguments optionnels
- 1 × lire et écrire fichiers texte
- 1 × threads et locks
- Manipulation d'images: concepts à savoir (possible dans QCM); pas dans partie pratique
- Deux exemples maintenant
- Exercices:
  - Davantage de temps pour les examens blancs
  - Nouvel exercice différent avec les thèmes ci-dessus

# Tâche 1: minigestionnaire de bibliothèque

---

```
library = Library({})

# Adding books
library.add_book(Book(title="1984", author="George Orwell", isbn="2345678901"))
library.add_book(Book(title="To Kill a Mockingbird", author="Harper Lee", isbn="1234567890"))

# Displaying books
library.display_books()

# Borrowing a book
print(library.borrow_book("1234567890"))
print(library.borrow_book("1234567890")) # Trying to borrow again

# Returning a book
print(library.return_book("1234567890"))
print(library.return_book("1234567890")) # Returning again

# Displaying books again
library.display_books(sort_by=lambda book: book.isbn)
```

# Tâche 2: minigestionnaire académique

---

```
school = School({})

# Adding students
school.add_student("S001", "Alice")
school.add_student("S002", "Bob")

# Recording grades
school.record_grade("S001", "Analyse", 5.0)
school.record_grade("S001", "ICC", 6.0)
school.record_grade("S002", "Analyse", 5.0)
school.record_grade("S002", "ICC", 5.5)

# Displaying student information
school.display_student("S001")
school.display_student("S002")
```