**KEY CONSTRUCTS OF LOOPS**

The key constructions required to create a loop are **initialization.** Initialization takes place prior to the beginning of the loop starting. This is the step where the coder can assign an initial value to the variable(s) that will control or be used in the loop. Another construct required to create a loop is the **loop condition.** The loop condition is what determines If the loop should continue or stop running. The loop conditions should be noted to be a Boolean expression.  Additionally, the **loop body**, which is a block of code that is executed consistently until the code in the loop condition is no longer factual. A very important construct within the loop is the **update expression or the increment/decrement** which is what continues the loop. Without this construct the coder runs the risk of an infinite loop.

**SCENARIOS**

***Infinite Loop***

I decided to utilize something that we are all familiar with. The school portal. The school portal can be used for several task such as students viewing their grades (which we all so eagerly do), enroll in course, drop courses, and exit the portal (just using the basics here for this example). The reason for the infinite loop being the best use for this example is because the coder is unknowing of how long the student will remain in the portal and therefore, the program needs to run until the student opts to exit the student portal.

grades = {"Math": "A", "English": "B"}

courses = ["Math", "English", "Science," "History"]

enrolled\_courses = ["Math", "English", "Science"]

while True:

    print("\n--- Student Course Management System ---")

    print("1. View Grades")

    print("2. Enroll in a Course")

    print("3. Drop a Course")

    print("4. Exit")

    choice = input("Enter your choice (1-4): ")

    if choice == "1":

        print("\nYour Grades:")

        for course, grade in grades.items():

            grade = grades.get(course, "N/A")

            print(f"{course}: {grade}")

    elif choice == "2":

        print("\nAvailable Courses to Enroll:")

        for course in courses:

            if course not in enrolled\_courses:

                print(f" - {course}")

        new\_course = input("Enter the course name to enroll: ")

        if new\_course in courses and new\_course not in enrolled\_courses:

            enrolled\_courses.append(new\_course)

            print(f"Enrolled in {new\_course}.")

        else:

            print("Invalid course name or already enrolled.")

    elif choice == "3":

        print("\nEnrolled Courses:")

        for course in enrolled\_courses:

            print(f"- {course}")

        drop\_course = input("Enter the course name to drop: ")

        if drop\_course in enrolled\_courses:

            enrolled\_courses.remove(drop\_course)

            print(f"Dropped {drop\_course}.")

        else:

            print("Invalid course name or not enrolled in that course.")

    elif choice == "4":

        print("Exiting the system. Goodbye!")

        break # Exit the infinite loop

    else:

        print("Invalid choice. Please eneter a number between 1 and 4.")

***For Loop***

The next example piggy backs off of the first example. Sending a notification to a list of students. For this example the assumption is that we already have the email address and name of each student stored within a list. This loop is best for this example due to our previous knowledge of how many students you’re dealing with, we are not performing different actions per student, and we are working with a list.

students = ["Christina", "Kiara", "Liam", "Nylah"]

message = "Welcome to Priciples of Programming!"

print("Sending notifications to students...")

for student in students:

    print(f"Hello {student}, {message}")

1. Be sure to provide specific details for each scenario that illustrate why the different types of loops are required.
2. Include a brief pseudocode example to share that includes at least one loop.
3. Actively participate in this discussion by providing constructive feedback on the scenarios and details posted by your peers.