UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Colegio de Ciencias y Humanidades plantel Azcapotzalco

Diplomado: "La importancia de los lenguajes de programación de última generación como herramienta interdisciplinaria que desarrolle el pensamiento crítico y creativo en el CCH"

Objetivo general: Presentar un panorama general de los lenguajes de programación de última generación mediante el estudio teórico-práctico de las herramientas y estrategias de aprendizaje de programación que éstos conllevan.

Objetivos particulares:

- Plantear un acercamiento práctico a la enseñanza-aprendizaje de lenguajes de programación de última generación a nivel bachillerato con la finalidad de identificar la importancia de esta actividad como herramienta tecnológica y multidisciplinaria.
- Identificar el impacto de la programación en el perfil del egresado del CCH cuyo propósito consiste en abrir el diálogo sobre la adopción de esta actividad como herramienta tecnológica necesaria.
- Presentación y enseñanza de dos lenguajes de programación de alto nivel: Python y Julia. El propósito es motivar a los profesores del colegio a la adopción de nuevas técnicas y herramientas de programación.
- Discutir sobre la programación como herramienta de trabajo docente en distintas áreas disciplinarias lo que permitirá identificar necesidades particulares en las que la programación puede ser de utilidad.
- Ofrecer un enfoque actualizado para la revisión y evaluación de los programas de estudios de las asignaturas que utilizan herramientas computacionales y el planteamiento de nuevos retos.
- Diseño de una propuesta didáctica utilizando Python y/o Julia mediante la cual el participante valide las herramientas, técnicas y estrategias obtenidas.

Participante: José Luis Méndez Becerril

"Inventario piloto de actividades computacionales (en Python) que complementen aprendizajes de la asignatura Matemáticas 1."

Notas:

- El entregable de este diplomado se concibió como un "inventario piloto" de "actividades computacionales" que los profesores de Matemáticas 1 puedan integrar a sus propias estrategias didácticas. Por eso no se presenta este avance como una estrategia didáctica.
- Por el momento, no se considera que estas actividades las programen los alumnos, aunque se podría, dependiendo de los tiempos y de los planes didácticos de cada docente.
- ¿Porqué Python?: Por su difusión.
- ¿Porqué Julia no?: Por tener una menor difusión y porque no se requiere capacidad de cómputo.
- ¿Porqué JAVA no?: ¡Porque me reprueban!

El espíritu de este entregable es provocar el interés de profesores por crear un inventario de actividades computacionales que complementen los objetivos de los aprendizajes de los Programas de Estudio.

Se podría crear un "grupo de trabajo" o "seminario" que se dedique a esto...

Unidad 1. El significado de los números y sus operaciones básicas

Aprendizaje 1: Comprende el significado de los números reales.

Temática: Significado de los números racionales (enteros y no enteros) e irracionales.

Actividad 1: Contar objetos.

Reflexionar la utilidad de los números para contar. Se podría discutir los diferentes números.

```
import random # Importo la librería random

print('¿Para qué sirven los números naturales?')
# Establecer el número de objetos a partir de un número aleatorio
cantidad = random.randint(1, 20) # De 1 a 20 objetos aleatorios se
pueden generar AQUÍ SE DEFINE EL NumMAX DE OBJETOS
# Generar la lista con los objetos (emoji de manzana)
objetos = ['🍏'] * cantidad # Uso la imagen de una manzana y la
multiplico por la cantidad aleatoria
print(' '.join(objetos)) #Imprime los objetos en fila suparados por '
'
```

```
# Obtener el dato del usuario
conteo_usuario = float(input('¿Cuántas manzanas hay?: '))
# Checar si el conteo del usuario es correcto
if conteo_usuario == cantidad:
    print("Correcto! Buen trabajo!")
elif conteo_usuario==0:
    print("¿Qué no ves alguna manzana?")
elif conteo_usuario<0:
    print("¿De verdad crees que puede haber una cantidad negativa de
manzanas?. ¿No sabes contar?")
elif isinstance(conteo_usuario,float):
    print("¿De verdad estás viendo que hay pedazos de manzanas?,
¿Tiene lógica tu pensamiento?")
else:
    print(f"Incorrecto!. Hay {cantidad} manzanas. ¿No sabes contar?")
print("\nEntonces, ¿Para qué sirven los números naturales?: ")
```

```
¿Para qué sirven los números naturales?
🍏 🍏 🍏
¿Cuántas manzanas hay?: 0
¿Qué no ves alguna manzana?

Entonces, ¿Para qué sirven los números naturales?:
```

¿Qué más se puede contar o medir?: dinámicas de poblaciones, objetos celestes, flujos de comercio, fenómenos temporales.

Otras magnitudes no absolutas:

- Sonido (decibeles), se requiere el establecimiento de un "cero absoluto" que es el umbral de audición humana, definido en 0 [dB].
- El pH es una medida de la acidez o bascicidad. Se define como el logaritmo negativo de la concentración de iones hidrógeno. La escala pH va de 0 a 14, siendo el 7 el valor neutro. Un valor >7 indica una solución ácida, mientras que un valor <7 indica una solución básica.
- El potencial eléctrico es una medida de la energía potencial eléctrica por unidad de carga en un punto dado en el espacio. Se mide en Volts y es una magnitud relativa al punto de referencia, el cual usualmente se define como un punto de cero potencial eléctrico.

Aprendizaje 1: Comprende el significado de los números reales.

Temática: Significado de los números racionales (enteros y no enteros) e irracionales.

Actividad 2: Comprender números racionales en la recta numérica

Comprensión de los números racionales mediante el hecho de dividir la unidad de referencia arbitraria un número "q" de veces para contar "p" de estas veces, y así poder ubicar en la recta numérica el número racional p/q. El valor obtenido (en decimal, por ejemplo), significa que los

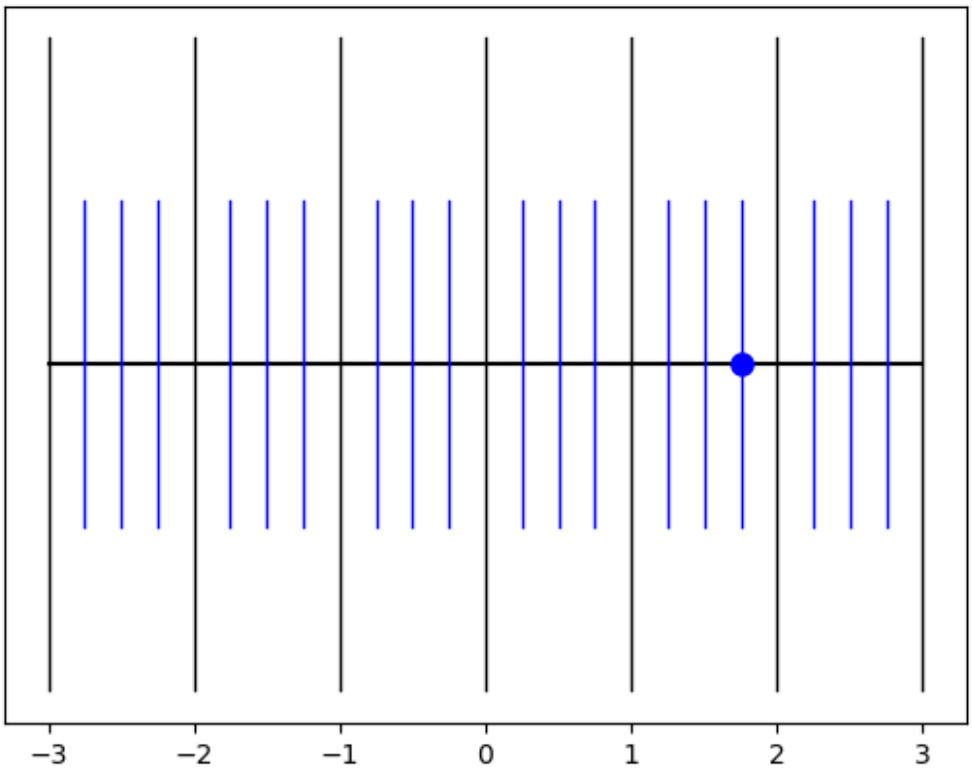
dos números enteros están relacionados mediante tal valor N. Por lo tanto "p" está contenido N veces en "q".

```
import matplotlib.pyplot as plt # Importo el método pyplot de la librería matplotlib

print("Ubicaremos números racionales entre -3 y 3 en la recta numérica")
# Aquí se establecen los extremos de la recta numérica
inicio = -3
fin = 3
# Preguntar al usuario en cuántas partes quiere divir la unidad de referencia
q = int(input('¿En cuántas partes iguales quieres divir la unidad de referencia (q)?: '))
# Crear una figura y el eje X
figura, ejeX = plt.subplots()
# Graficar la recta numérica
ejeX.plot([inicio, fin], [0, 0], 'k-')
# Graficar las divisiones en la recta numérica
for num in range(inicio * q, fin * q + 1):
    x = num / q
    if num % q == 0:
        ejeX.plot([x, x], [-0.2, 0.2], 'k-', linewidth=1)
        ejeX.annotate(str(int(x)), (x, -0.5))
    else:
        ejeX.plot([x, x], [-0.1, 0.1], 'b-', linewidth=1)
# Solicitar el numerador del número racional
p = int(input(f'¿Cuántas partes quieres contar (entre {inicio * q} y {fin * q}), (p): '))
numero_racional = p / q
# Graficar el número racional ingresado por el usuario
ejeX.plot(numero_racional, 0, 'bo', markersize=8)
ejeX.annotate(f'{p}/{q}', (numero_racional, 0.5))
# Remover el eje Y
ejeX.yaxis.set_visible(False)
print("\nLa ubicación en la recta numérica de %d/%d es:"% (p,q))
# Mostrar la gráfica
plt.show()
```

```
Ubicaremos números racionales entre -3 y 3 en la recta numérica
¿En cuántas partes iguales quieres divir la unidad de referencia (q)?: 4
¿Cuántas partes quieres contar (entre -12 y 12), (p): 7

La ubicación en la recta numérica de 7/4 es:
```



Aprendizaje 2 Usa correctamente las diversas simbolizaciones de un número racional, transitando entre sus equivalencias (cuando sea necesario) en problemas puramente aritméticos y en contexto.

Actividad 3: ¿Soy un número racional?

```
from fractions import Fraction # Importo la clase Fraction de la librería fractions para trabajar con fracciones

numero = float(input("Ingresa un número decimal: ")) # Obtengo dato del usuario
fraccion = Fraction(numero).limit_denominator() # Uso la clase Fraction para crear un objeto Fraction que represente un número racional y luego llamo al método limit_denominator() en el objeto para obtener una fracción
#El método limit_denominator() obtiene la representación racional más próxima

if fraccion.numerator == numero: # Si el número es igual al numerador, es un número entero
    print(f"{numero} es un número entero.")
elif abs(fraccion.denominator) > 10**6: # Si el valor absoluto del denominador de fraccion es mayor a 1M...
    print(f"{numero} es un número irracional.")
```

```
else:
    print(f"{numero} es un número racional y puede ser expresado como {fraccion} en su forma simplificada. ¿Porqué?")
```

```
Ingresa un número decimal: 3.3
3.3 es un número racional y puede ser expresado como 33/10 en su forma simplificada. ¿Porqué?
```

Aprendizaje 3: Compara dos cantidades haciendo uso de las representaciones de un número racional

Actividad 3: Adivina el número que estoy pensando

Comprensión de la posición de números en la recta numérica (mayores a la derecha, menores a la izquierda).

```
import random

def adivina_numero():
    numero = random.randint(-10,10) #Aquí se define el intervalo en el que estará el número a adivinar
    intentos = 0
    while True:
        tuintento = int(input("¿Qué número entero entre -10 y 10 crees que estoy pensando?: ")) #Aquí es importante la consistencia del intervalo impreso
        intentos += 1
        if tuintento == numero:
            print(f"Bien! Adivinaste el número entero en {intentos} intentos.")
            break
        elif tuintento < numero:
            print("El número entero que estoy pensando está más a la derecha en la recta numérica. ¿Soy mayor o menor?")
        else:
            print("El número entero que estoy pensando está más a la izquierda en la recta numérica. ¿Soy mayor o menor?")

print("¿Lograrás adivinar el número entero que estoy pensando?")
adivina_numero()
```

```
¿Lograrás adivinar el número entero que estoy pensando?
¿Qué número entero entre -10 y 10 crees que estoy pensando?: 5
El número entero que estoy pensando está más a la izquierda en la recta numérica. ¿Soy mayor o menor?
¿Qué número entero entre -10 y 10 crees que estoy pensando?: 3
El número entero que estoy pensando está más a la izquierda en la recta numérica. ¿Soy mayor o menor?
```

Aprendizaje 4: Opera correctamente con los números racionales (enteros y no enteros), en los casos de una sola operación y una secuencia de operaciones

Actividad 4: Calculadora aritmética básica con números racionales

Reflexionar con los alumnos los algoritmos para las operaciones básicas con números racionales.

```
# ¿Agregar entorno gráfico?

import math
from fractions import Fraction

def aritm_fracc():
    while True:
        operacion = input("¿Qué operación aritmética básica entre dos
números racionales quieres realizar?: (+, -, *, /) o ingresa 's' para
salir: ")
        if operacion == 's':
            break
        fraccion1 = Fraction(input("Ingresa el primer número racional
en la forma p/q: "))
        fraccion2 = Fraction(input("Ingresa el segundo número racional
en la forma p/q: "))
        if operacion == '+':
            mcm = fraccion1.denominator * fraccion2.denominator //
math.gcd(fraccion1.denominator, fraccion2.denominator)
            print(f"El mínimo común múltiplo de los denominadores es:
{mcm}")
            f1 = Fraction(fraccion1.numerator * (mcm //
fraccion1.denominator), mcm)
            f2 = Fraction(fraccion2.numerator * (mcm //
fraccion2.denominator), mcm)
            print(f"{fraccion1} = {f1} y {fraccion2} = {f2}")
            resultado = f1 + f2
            print(f"{f1} + {f2} = {resultado}")
            print("Bye! Bye!")
            break
        elif operacion == '-':
            mcm = fraccion1.denominator * fraccion2.denominator //
math.gcd(fraccion1.denominator, fraccion2.denominator)
            print(f"El mínimo común múltiplo de los denominadores es:
{mcm}")
            f1 = Fraction(fraccion1.numerator * (mcm //
fraccion1.denominator), mcm)
            f2 = Fraction(fraccion2.numerator * (mcm //
fraccion2.denominator), mcm)
            print(f"{fraccion1} = {f1} y {fraccion2} = {f2}")
```

```
            resultado = f1 - f2
            print(f"{f1} - {f2} = {resultado}")
            print("Bye! Bye!")
            break
        elif operacion == '*':
            resultado = fraccion1 * fraccion2
            print("En la multiplicación de números racionales no
involucramos al mcm de los denominadores.")
            print(f"{fraccion1} * {fraccion2} = {resultado}")
            print("     1. Primero obtienes el signo del resultado
haciendo uso de las leyes de los signos.")
            print("     2. Después, multiplica numeradores y el
resultado será el numerador.")
            print("     3. Por último, multiplica denominadores y el
resultado será el denominador.")
            print("Siempre simplificarás la fracción si es posible.")
            print("Bye! Bye!")
            break
        elif operacion == '/':
            resultado = fraccion1 / fraccion2
            print("En la división de números racionales no
involucramos al mcm de los denominadores.")
            print(f"{fraccion1} / {fraccion2} = {resultado}")
            print("     1. Primero obtienes el signo del resultado
haciendo uso de las leyes de los signos.")
            print("     2. Después, multiplicas los números externos
p1 y q2 y el resultado será el numerador")
            print("     2. Después, multiplicas los números internos
q1 y p1 y el resultado será el denominador")
            print("Bye! Bye!")
            break
        else:
            print("Selección inválida. ¿Qué no pones antención?")
            print("Bye! Bye!")
            continue

print("Programa para realizar operaciones aritméticas básicas entre
dos números racionales, ambos de la forma p/q")
aritm_fracc()
```

```
Programa para realizar operaciones aritméticas básicas entre dos
números racionales, ambos de la forma p/q
¿Qué operación aritmética básica entre dos números racionales quieres
realizar?: (+, -, *, /) o ingresa 's' para salir: +
Ingresa el primer número racional en la forma p/q: 1/2
Ingresa el segundo número racional en la forma p/q: 4/2
El mínimo común múltiplo de los denominadores es: 2
1/2 = 1/2 y 2 = 2
1/2 + 2 = 5/2
Bye! Bye!
```

```
# Dibujar una línea y adivinar su longitud para reflexionar sobre la
# utilidad de los números enteros para medir, contar

import random
from PIL import Image, ImageDraw

def dibujar_linea(length, direction):
    # Crear una imagen con fondo blanco
    imagen = Image.new('RGB', (400, 100), 'white')
    dibujar = ImageDraw.Draw(imagen)

    # Dibujar la línea
    if direction == 'positiva':
        dibujar.line((50, 50, 50 + length, 50), fill='black', width=3)
        dibujar.polygon([(50 + length - 10, 40), (50 + length, 50),
(50 + length - 10, 60)], fill='black')
    else:
        dibujar.line((350, 50, 350 - length, 50), fill='black',
width=3)
        dibujar.polygon([(350 - length + 10, 40), (350 - length, 50),
(350 - length + 10, 60)], fill='black')

    # Mostrar la linea
    imagen.show()

def adivinar_longitud_linea():
    # Genera una linea con valores de longitud y dirección aleatorios
    longitud_linea = random.randint(1, 300) # La longitud es un valor
entero aleatorio entre 1 y 300
    linea_direccion = random.choice(['positiva', 'negativa']) # Se
elige aleatoriamente la dirección

    # Dibujar la linea
    dibujar_linea(longitud_linea, linea_direccion)

    # Calcular el intervalo para el valor de la longitud de la línea
    extremo_inferior = max(1, longitud_linea - 20)
    extremo_superior = min(300, longitud_linea + 20)

    # Mostrar el intervalo en el que se encuentra la longitud de la
línea
    print(f'La longitud de la linea está entre {extremo_inferior} y
```

```
{extremo_superior}.')

    # Iterar hasta que el usuario adivine la longitud de la línea
    while True:
        # Preguntar al usuario por la longitud de la línea
        intento = int(input('¿Cuál es la longitud entera de esta
linea?: '))

        # Evaluar si el intento es correcto
        if intento == longitud_linea:
            print('Correcto!')
            break
        elif intento < longitud_linea:
            print('Mi longitud es un número entero mayor. ¿Mi longitud
está a la derecha o a la izquierda en la recta numérica?')
        else:
            print('Mi longitud es un número entero menor. ¿Mi longitud
está a la derecha o a la izquierda en la recta numérica?')

# Llamar la función
adivinar_longitud_linea()
```