IP

Introducción a la Programación

CONCEPTOS GENERALES – VERSIÓN 1.0

PROFESOR: VICTOR HUGO CONTRERAS

Contenido

Introducción a la Programación

Unidad 1: Conceptos Básicos y Primeros Programas

- Introducción a la programación y su importancia.
- Tipos de lenguajes de programación (compilados e interpretados).
- **Diagramas de Jackson**: Introducción para representar la estructura de un programa simple.
- Introducción a **PSint**: Pseudocódigo básico y lógica secuencial.
- Primeros programas en **Python**: Entrada/salida básica.
- Ejercicio: Crear un diagrama de Jackson para un programa simple, implementarlo en PSint y luego en Python.

Unidad 2: Tipos de Datos, Operadores y Uso Combinado

- Tipos de datos en Python: enteros, flotantes, cadenas, booleanos.
- Variables y operadores en Python: aritméticos, relacionales y lógicos.
- **Diagramas de Jackson**: Representación de manipulación de datos y operaciones.
- **PSint**: Programación en pseudocódigo con variables y operadores.
- Implementación en **Python**: Operaciones con tipos de datos y operadores.
- Ejercicio: Planificar con diagramas de Jackson, escribir el pseudocódigo en PSint y ejecutar la solución en Python.

Unidad 3: Estructuras de Control

- Estructuras condicionales en Python: if, else, elif.
- Bucles: for, while, break, continue.
- **Diagramas de Jackson**: Representación visual de decisiones y bucles.
- **PSint**: Escritura de pseudocódigo que utilice estructuras de control.
- **Python**: Implementación de condicionales y bucles.
- Ejercicio: Usar diagramas de Jackson para planificar el flujo, escribirlo en PSint y luego traducirlo a Python.

Unidad 4: Funciones y Modularización

- Definición y uso de funciones en Python: parámetros, argumentos y valores de retorno.
- **Diagramas de Jackson**: Diseño de un programa modular.
- **PSint**: Estructuración de problemas en funciones.
- **Python**: Creación de funciones y módulos en código.
- Ejercicio: Crear diagramas de Jackson para descomponer un problema, escribir las funciones en PSint y luego implementarlas en Python.

Unidad 5: Estructuras de Datos Básicas (Listas y Cadenas)

- Introducción a listas (vectores) y su uso en Python.
- Manipulación de cadenas en Python.
- **Diagramas de Jackson**: Planificación de algoritmos utilizando listas.
- **PSint**: Escritura de pseudocódigo que manipule listas y cadenas.
- **Python**: Implementación de operaciones con listas y cadenas.
- Ejercicio: Diseñar con diagramas de Jackson, resolver en PSint y llevar a cabo en Python.

Evaluación

- **Evaluaciones parciales**: Usar diagramas de Jackson para planificación, escribir pseudocódigo en PSint y luego implementar la solución en Python.
- **Proyecto final**: Desarrollar un programa completo desde el diseño con Jackson, el pseudocódigo en PSint y la implementación en Python.

Desarrollo de conceptos

Unidad 1: Conceptos Básicos y Primeros Programas

Introducción: ¡Bienvenidos al mundo de la programación! Hoy comenzamos un viaje donde aprenderemos a comunicarnos con las computadoras y a crear programas que realicen tareas útiles. La programación es fundamental en nuestra vida diaria; desde aplicaciones móviles hasta sistemas de control en automóviles, todo se basa en código.

Conceptos Clave:

1. **Programación**: Definimos la programación como el proceso de escribir instrucciones que una computadora puede entender. Cada tarea que queremos que realice la computadora necesita ser codificada.
2. **Lenguaje de programación**: Un lenguaje que utilizamos para escribir instrucciones. Hoy nos enfocaremos en Python, conocido por su simplicidad y legibilidad.

Actividades:

1. **Presentación de la Programación**:
    - Comencemos con una discusión. ¿Qué aplicaciones conocen que utilizan programación? (Ejemplos: redes sociales, videojuegos, plataformas de streaming).
    - Reflexionemos sobre cómo estas aplicaciones impactan nuestras vidas.
2. **Diagramas de Jackson**:
    - **Explicación**: Los Diagramas de Jackson son herramientas visuales que nos ayudan a planificar la lógica de nuestros programas. Nos permiten descomponer problemas complejos en partes más manejables.
    - **Ejercicio**: En grupos, elijan una tarea simple, como calcular el área de un rectángulo, y dibujen un diagrama que represente los pasos necesarios.
3. **Introducción a PSint**:
    - **Explicación**: PSint es un lenguaje de pseudocódigo que nos ayuda a pensar en la lógica de un programa sin preocuparnos por la sintaxis

exacta de Python. Es un paso intermedio entre el pensamiento y el código real.

- **Ejercicio**: Escriban un pseudocódigo que sume dos números. Luego, compartan sus pseudocódigos y discutan las diferencias.

4. **Programación en Python**:
    - **Explicación**: Ahora, pasemos a Python. Mostramos cómo instalar Python y un entorno de desarrollo.
    - **Ejercicio**: Escriban un programa en Python que reciba dos números y muestre su suma. (Demuestra en clase cómo se hace).

Unidad 2: Tipos de Datos, Operadores y Uso Combinado

Introducción: En esta unidad, profundizaremos en los tipos de datos y operadores, elementos esenciales para cualquier programa. Comprender cómo manejar diferentes tipos de datos es crucial para construir aplicaciones efectivas.

Conceptos Clave:

1. **Tipos de Datos**: Python ofrece varios tipos de datos, entre ellos:
    - **int**: Números enteros.
    - **float**: Números de punto flotante (decimales).
    - **str**: Cadenas de texto.
    - **bool**: Valores booleanos (verdadero o falso).
2. **Operadores**: Son símbolos que indican operaciones a realizar. Los más comunes son:
    - **Aritméticos**: +, -, *, /
    - **Comparación**: ==, !=, <, >
    - **Lógicos**: and, or, not

Actividades:

1. **Tipos de Datos**:
    - **Explicación**: Veamos ejemplos de cada tipo de dato. ¿Qué sucede si intentamos realizar operaciones con diferentes tipos?
    - **Ejercicio**: Presente una lista de valores y pida a los estudiantes que identifiquen su tipo de dato.
2. **Diagramas de Jackson**:
    - **Ejercicio**: Usen Diagramas de Jackson para representar la operación de cálculo del área de un triángulo. Pregunten a los estudiantes cómo dividirían este problema.
3. **PSint**:
    - **Explicación**: Hable sobre cómo el pseudocódigo puede ayudarnos a planificar operaciones.

- **Ejercicio**: Escriban un pseudocódigo que calcule el promedio de tres números. Luego, revisen juntos los pseudocódigos.

4. **Implementación en Python**:
    - **Ejercicio**: Desarrollen un programa en Python que reciba tres números y calcule su promedio.
    - **Discusión**: Comparen las soluciones y analicen errores comunes.

Unidad 3: Estructuras de Control

Introducción: Las estructuras de control son fundamentales para la toma de decisiones en la programación. Hoy aprenderemos sobre las estructuras condicionales y los bucles, que nos permiten repetir acciones y tomar decisiones basadas en condiciones.

Conceptos Clave:

1. **Estructuras Condicionales**: Nos permiten ejecutar un bloque de código solo si se cumple una condición. Las instrucciones más comunes son:
    - if, else, y elif.
2. **Bucles**: Permiten repetir un bloque de código varias veces. Los más usados son:
    - for: Repite un bloque un número específico de veces.
    - while: Repite mientras una condición sea verdadera.

Actividades:

1. **Estructuras Condicionales**:
    - **Ejemplo**: Presenta un escenario: si un estudiante aprueba (más de 60), muestra "Aprobado", de lo contrario "Reprobado".
    - **Ejercicio**: Pide a los estudiantes que escriban un programa que clasifique a los estudiantes según su calificación.
2. **Diagramas de Jackson**:
    - **Ejercicio**: Realicen un diagrama para el flujo del juego "Adivina el número", mostrando cómo toman decisiones en el programa.
3. **Bucles**:
    - **Explicación**: Muestre cómo funcionan los bucles con ejemplos prácticos.
    - **Ejercicio**: Escribir un programa que imprima los números del 1 al 10.
4. **PSint y Python**:
    - **Ejercicio**: Pidan a los estudiantes que escriban pseudocódigo para calcular la suma de los números del 1 al 100 usando un bucle.

- **Implementación**: Programar en Python y discutir errores comunes.

Unidad 4: Funciones y Modularización

Introducción: Las funciones son bloques de código que nos permiten modularizar y reutilizar código. En esta unidad, exploraremos cómo definir y usar funciones para mejorar la legibilidad y eficiencia de nuestros programas.

Conceptos Clave:

1. **Funciones**: Un bloque de código que realiza una tarea específica. Las funciones pueden aceptar parámetros y devolver valores.
2. **Modularización**: La práctica de dividir un programa en partes más pequeñas y manejables, lo que facilita la lectura y el mantenimiento.

Actividades:

1. **Introducción a Funciones**:
    - **Ejemplo**: Definir una función simple que sume dos números.
    - **Ejercicio**: Pidan a los estudiantes que creen una función que calcule el factorial de un número.
2. **Diagramas de Jackson**:
    - **Ejercicio**: Crear un diagrama que muestre la lógica de un programa modular que calcula el área de diferentes formas.
3. **PSint**:
    - **Ejercicio**: Escribir pseudocódigo para una función de calculadora que maneje operaciones básicas.
    - **Implementación**: Programar la función en Python y probarla.
4. **Ejercicios de Implementación**:
    - **Ejercicio**: Modifiquen su calculadora para que acepte múltiples operaciones y muestre resultados intermedios.

Unidad 5: Estructuras de Datos Básicas (Listas y Cadenas)

Introducción: Las listas y cadenas son estructuras de datos fundamentales en Python. Hoy aprenderemos cómo utilizarlas para almacenar y manipular datos de manera efectiva.

Conceptos Clave:

1. **Listas**: Estructuras que pueden contener múltiples elementos. Son mutables y se pueden modificar después de su creación.
2. **Cadenas**: Secuencias de caracteres que son inmutables. Puedes manipularlas con diferentes métodos.

Actividades:

1. **Introducción a Listas**:
    - **Ejemplo**: Crear una lista de frutas y mostrar cómo acceder a sus elementos.
    - **Ejercicio**: Escribir un programa que almacene nombres de estudiantes y los imprima.
2. **Cadenas**:
    - **Ejemplo**: Presentar cómo manipular cadenas (corte, concatenación).
    - **Ejercicio**: Crear un programa que cuente la cantidad de vocales en una cadena.
3. **Diagramas de Jackson**:
    - **Ejercicio**: Planificar un algoritmo que busque un elemento en una lista, representándolo con un diagrama.
4. **PSint y Python**:
    - **Ejercicio**: Escribir pseudocódigo para manipular listas y cadenas.
    - **Implementación**: Crear un programa que reciba una lista y devuelva la lista ordenada.

Recomendaciones Generales para la Enseñanza

- **Enfoque Interactivo**: Involucrar a los estudiantes en discusiones y actividades prácticas.
- **Ejemplos del Mundo Real**: Relacionar conceptos de programación con aplicaciones del día a día.
- **Retroalimentación Continua**: Proporcionar comentarios constructivos sobre las tareas y ejercicios.
- **Utiliza Recursos Visuales**: Los diagramas y visualizaciones pueden ayudar a los estudiantes a comprender mejor los conceptos.

Ejercicios

Unidad 1: Conceptos Básicos y Primeros Programas

1. **Hola Mundo**: ¡Escribe un programa que imprima “¡Hola, Mundo!".
2. **Suma de Números**: Solicita dos números al usuario y muestra su suma.
3. **Restar Números**: Solicita dos números y muestra la diferencia.
4. **Multiplicar Números**: Solicita dos números y muestra el producto.
5. **Dividir Números**: Solicita dos números y muestra el cociente.
6. **Intercambio de Variables**: Escribe un programa que intercambie los valores de dos variables.
7. **Conversión de Grados**: Pide al usuario una temperatura en grados Celsius y convierte a Fahrenheit.
8. **Área de un Rectángulo**: Pide la base y la altura de un rectángulo y calcula su área.
9. **Diagrama de Jackson**: Crea un diagrama para calcular el área de un círculo y luego programa en Python.
10. **Dibujo de Figuras**: Escribe un programa que dibuje un triángulo, cuadrado y círculo usando texto.
11. **Comentarios en Código**: Agrega comentarios a tu programa "Hola Mundo" explicando cada línea.
12. **Variable Tipo String**: Declara una variable de tipo cadena y muestra su longitud.
13. **Números Impares**: Escribe un programa que imprima todos los números impares del 1 al 50.
14. **Contador de Vocales**: Pide una cadena y cuenta cuántas vocales tiene.
15. **Condiciones Simples**: Escribe un programa que muestre si un número es par o impar.
16. **Pedir Nombre**: Crea un programa que pida al usuario su nombre y lo salude.
17. **Tablas de Multiplicar**: Imprime la tabla de multiplicar del 5.

18. **Saludo Personalizado**: Escribe un programa que pregunte el nombre y edad del usuario y muestre un saludo personalizado.
19. **Conversión de Unidades**: Convierte metros a kilómetros y muestra el resultado.
20. **Dibujo de Estrellas**: Imprime un triángulo de estrellas con un número de filas dado por el usuario.

Unidad 2: Tipos de Datos, Operadores y Uso Combinado

1. **Tipos de Datos**: Declara variables de tipo int, float, str y bool, e imprime sus tipos.
2. **Operaciones Aritméticas**: Escribe un programa que reciba dos números y muestre su suma, resta, multiplicación y división.
3. **Promedio de Notas**: Pide cinco notas al usuario y calcula el promedio.
4. **Comparaciones**: Compara dos números ingresados por el usuario e indica cuál es mayor.
5. **Uso de Módulos**: Escribe un programa que use el operador módulo para determinar si un número es par o impar.
6. **Concatenación de Cadenas**: Pide dos cadenas y muéstralas concatenadas.
7. **Cálculo de Área**: Calcula el área de un círculo a partir de su radio, usando la fórmula A = π * r^2.
8. **Conversión de Moneda**: Convierte una cantidad en dólares a pesos utilizando un tipo de cambio fijo.
9. **Uso de la Función len**: Pide una cadena y muestra su longitud usando len().
10. **Búsqueda de Subcadenas**: Pregunta al usuario por una cadena y una subcadena, y verifica si esta última está presente.
11. **Contador de Caracteres**: Cuenta la cantidad de caracteres en una cadena ingresada por el usuario.
12. **Condiciones Compuestas**: Escribe un programa que evalúe si un número es positivo y par.

13. **Cálculo de Descuentos**: Pide un precio y un porcentaje de descuento y muestra el precio final.
14. **Cálculo de IMC**: Solicita peso y altura y calcula el Índice de Masa Corporal (IMC).
15. **Cálculo de Notas**: Pide la calificación de un estudiante y muestra "Aprobado" o "Reprobado".
16. **Contador de Palabras**: Cuenta cuántas palabras hay en una cadena ingresada.
17. **Mayor de Tres Números**: Escribe un programa que pida tres números y muestre el mayor.
18. **Operaciones Lógicas**: Crea un programa que verifique si un número está entre 1 y 10.
19. **Escalera de Números**: Imprime los números del 1 al 20, indicando si son pares o impares.
20. **Cálculo de Intereses**: Pide un capital inicial, un interés y un tiempo, y calcula el interés simple.

Unidad 3: Estructuras de Control

1. **Calificación Estudiante**: Clasifica la calificación de un estudiante en "Excelente", "Buena", "Regular" o "Mala".
2. **Juego de Adivinanza**: Crea un juego en el que el usuario adivine un número entre 1 y 50, usando bucles y condiciones.
3. **Suma de Números**: Escribe un programa que sume todos los números del 1 al 100 usando un bucle.
4. **Tablas de Multiplicar**: Genera la tabla de multiplicar de un número ingresado por el usuario.
5. **Contador de Números**: Cuenta cuántos números pares hay entre 1 y 50.
6. **Dibujo de Pirámide**: Pide un número y dibuja una pirámide de estrellas.
7. **Bucle Infinito**: Crea un programa que pida números al usuario hasta que se ingrese un 0.

8. **Cálculo de Factorial**: Implementa el cálculo del factorial de un número usando un bucle.
9. **Contador de Vocales**: Pide una cadena y cuenta cuántas vocales tiene usando un bucle.
10. **Números Primos**: Escribe un programa que muestre todos los números primos entre 1 y 100.
11. **Contador de 10 en 10**: Imprime los números del 1 al 100 contando de 10 en 10.
12. **Juego de Piedra, Papel o Tijera**: Implementa un juego simple donde el usuario juega contra la computadora.
13. **Gráfica de Notas**: Crea un programa que solicite notas de estudiantes hasta que se ingrese una nota negativa, y luego muestre la nota promedio.
14. **Adivinanza de Números**: Realiza un juego donde la computadora elige un número y el usuario intenta adivinarlo.
15. **Cálculo de Suma**: Suma todos los números impares entre 1 y 100.
16. **Calificaciones de Grupos**: Pregunta por las calificaciones de varios grupos y calcula el promedio.
17. **Cálculo de Descuentos**: Pide el precio de un producto y aplica un descuento según el rango de precio.
18. **Simulación de Cajero**: Escribe un programa que simule un cajero automático (consultar saldo, retirar y depositar).
19. **Conversor de Temperatura**: Convierte temperaturas de Celsius a Fahrenheit y viceversa.
20. **Contador de Palabras**: Crea un programa que pida una frase y cuente cuántas palabras hay.

Unidad 4: Funciones y Modularización

1. **Función de Suma**: Crea una función que sume dos números y retorne el resultado.
2. **Función de Área**: Implementa una función que calcule el área de un triángulo dados base y altura.
3. **Factorial**: Escribe una función que calcule el factorial de un número.
4. **Verificación de Palíndromos**: Crea una función que determine si una palabra es un palíndromo.
5. **Conversión de Unidades**: Implementa funciones para convertir entre diferentes unidades (metros a kilómetros, etc.).
6. **Generador de Números Aleatorios**: Crea una función que genere un número aleatorio entre dos límites.
7. **Calculadora**: Diseña una función que realice operaciones básicas (suma, resta, multiplicación, división).
8. **Verificación de Primos**: Implementa una función que verifique si un número es primo.
9. **Ordenar Lista**: Crea una función que reciba una lista y la ordene.
10. **Contar Vocales**: Escribe una función que cuente las vocales en una cadena.
11. **Contar Caracteres**: Crea una función que cuente la cantidad de caracteres en una cadena.
12. **Calculadora de IMC**: Implementa una función que calcule el índice de masa corporal (IMC).
13. **Funciones Recursivas**: Escribe una función recursiva para calcular la serie de Fibonacci.
14. **Dibujo de Estrellas**: Crea una función que dibuje un cuadrado de estrellas.
15. **Conversión de Temperatura**: Escribe una función para convertir entre Celsius y Fahrenheit.
16. **Comparar Cadenas**: Crea una función que compare dos cadenas e indique cuál es mayor lexicográficamente.

17. **Filtrar Números**: Implementa una función que filtre números pares de una lista.
18. **Contador de Palabras**: Crea una función que cuente cuántas palabras hay en una frase.
19. **Calculadora Modular**: Diseña una calculadora que use funciones para realizar operaciones matemáticas.
20. **Generador de Contraseñas**: Implementa una función que genere una contraseña aleatoria de una longitud dada.

Unidad 5: Listas y Cadenas

1. **Manipulación de Listas**: Crea una lista con cinco elementos y muéstrala en la consola.
2. **Suma de Elementos**: Escribe un programa que calcule la suma de todos los elementos de una lista.
3. **Buscar en Lista**: Pide al usuario un número y verifica si está en una lista predefinida.
4. **Invertir Lista**: Crea una función que invierta el orden de los elementos de una lista.
5. **Listas de Nombres**: Pide una lista de nombres y muestra el que comienza con la letra 'A'.
6. **Contar Elementos**: Escribe un programa que cuente cuántas veces aparece un elemento en una lista.
7. **Unir Listas**: Crea dos listas y une sus elementos en una sola lista.
8. **Eliminar Duplicados**: Escribe un programa que elimine elementos duplicados de una lista.
9. **Promedio de Notas**: Pide al usuario que ingrese notas y almacénalas en una lista, luego calcula el promedio.
10. **Funciones con Listas**: Crea una función que reciba una lista y retorne su mayor y menor elemento.

11. **Cadenas en Lista**: Crea una lista de palabras y muestra solo aquellas que tienen más de 3 letras.
12. **Contar Caracteres en Cadena**: Pide una cadena y cuenta cuántas veces aparece una letra específica.
13. **Conversión de Cadenas**: Escribe un programa que convierta una cadena a mayúsculas y minúsculas.
14. **Lista de Compras**: Crea una lista de compras, permite al usuario agregar y eliminar elementos.
15. **Buscar Subcadena**: Pide una cadena y una subcadena, y verifica si la subcadena está en la cadena.
16. **Repetir Elementos**: Escribe un programa que pida un número y repita un mensaje esa cantidad de veces.
17. **Filtrar Números**: Crea una lista de números y muestra solo los que son mayores a 10.
18. **Ordenar Listas**: Escribe un programa que ordene una lista de números.
19. **Palíndromos en Lista**: Crea una lista de palabras y muestra cuáles son palíndromos.
20. **Cadenas con Formato**: Escribe un programa que formatee una cadena ingresada por el usuario para que tenga una primera letra en mayúscula.