

MINISTERIO DE EDUCACION PÚBLICA
DEPARTAMENTO DE ESPECIALIDADES TÉCNICAS



COLEGIO VOCACIONAL DE ARTES Y OFICIOS DE CARTAGO

PORTAFOLIO DE EVIDENCIAS
PORTAFOLIO DE EVIDENCIAS

Especialidad: Electrónica Industrial X Año

Subárea: Fundamentos de Electrónica

INFORMACIÓN GENERAL

Portafolio de Evidencias de Electrónica Industrial

Nivel: Decimo

Subárea: Fundamentos de Electrónica

Profesor a cargo: Lic. Olman Coto Alcázar

Nombre del Estudiante: Keneth Rojas Vindas
Sección: 10-A
Email: rojasvindaskeneth@gmail.com

Evidencia 1: Diagramas de flujo, Estructuras de control y primeros pasos en programación (Python).

Durante las clases virtuales recibidas en estas primeras semanas se habló sobre las máquinas y su funcionamiento.

Aquí los apuntes de esta clase:

- Maquinas eléctricas

Las maquinas eléctricas son dispositivos que pueden convertir energía mecánica en energía eléctrica o viceversa.

Se clasifican en:

-Estáticas: Un ejemplo es el transformador el cual es una maquina eléctrica estática, no posee partes móviles, en el cual, tomando el transformador como ejemplo, esta toma la energía eléctrica alterna y la transforma en energía eléctrica directa.

-Rotatorias: Como ejemplo para las maquinas eléctricas rotatorias, los generadores los cuales poseen partes móviles, son máquinas las cuales generalmente se encargan de transformar energía mecánica en energía eléctrica.

- Magnetismo:

- campo magnético: Es un campo de fuerza creado como consecuencia de la corriente, cuya fuerza se puede medir en Gauss (G) o tesla (T)

- H unidad ampere * metro (Am)

- Flujo magnético: Es la cantidad de líneas de campo magnético que atraviesan una superficie (S) en el espacio.

- Ø Unidad Weber (Wb)

Así como se ven los primeros pasos en programación aprendiendo definiciones sobre cosas útiles durante el día a día en Python.

Código fuente:

El código fuente de un programa informático se define como el conjunto de líneas de texto con los pasos que debe seguir la computadora para ejecutar un programa escrito en lenguaje específico de programación, en nuestro caso se utilizará el lenguaje complejo "Python".

Lenguaje de programación:

Es un sistema estructurado de comunicación el cual contiene en si conjuntos de símbolos, palabras claves, reglas semánticas y sintácticas, que permite a un programador escribir en un conjunto de órdenes, acciones y algoritmos para, de esa forma, crear programas que controlen el comportamiento físico y lógico de una máquina.

Se dividen en lenguajes de bajo nivel y de alto nivel.

Algoritmo:

Es una secuencia no ambigua, finita y ordenada de instrucciones, que se ejecutan en un orden establecido con el fin de resolver un problema.

Interprete y compilador:

Un compilador es un software que traduce un programa escrito en un lenguaje de programación de alto nivel (c/c++, java, Python, etc.) en lenguaje máquina. Un compilador generalmente genera lenguaje ensamblador primero y luego traduce el lenguaje ensamblador al lenguaje máquina.

Un dato importante que cabe mencionar es que la máquina como tal únicamente lee los números 0 y 1, por lo que, el compilador se encarga de traducir las palabras claves definidas por el mismo se traduzcan a la máquina como 0 y 1 como cantidades de corriente por decirlo de alguna forma y se pueda entender por la computadora o máquina lo que se le está enviando.

Depurador:

Es un programa utilizado para probar y eliminar los errores de otros programas

Palabras reservadas:

Las palabras reservadas, también conocidas como palabras claves, son aquellas tienen una función en el lenguaje de programación, por lo que en cada lenguaje las palabras reservadas cambian.

Las palabras reservadas para usar tipos de datos son:

- bool
- int
- char
- byte
- long
- double

Las estructuras de control:

- if
- else
- while
- for
- switch
- case
- break
- try
- return
- void

Lenguaje de programación Python:

Print(): esta función permite mostrar un texto en la pantalla.

Input(): esta función permite al usuario el ingreso de datos de distintos tipos.

Def: Define una función, es una sentencia ejecutable, para guardar líneas de código y ejecutarlas cuando sean necesarias.

Estructuras de control:

Son instrucciones que permiten romper una secuencialidad de la ejecución de un programa; esto significa que una estructura de control permite que se realicen unas instrucciones y omitir otras, de acuerdo a la evaluación de una condición, la cual siempre se define en si es verdadera o no.

Estas permiten modificar el flujo en el que se ejecutan las instrucciones de un programa.

Estas se clasifican en selectivas e iterativas:

Selectivas: Ejecutan un bloque de instrucciones u otro, o saltan a un subprograma o subrutina según se cumpla o no una condición, ejemplos de estas son if-elif.

Iterativas: Son aquellas que inician o repiten un bloque de instrucciones si se cumple una condición o mientras se llega a cumplir una condición. Ejemplos de estas son for-while.

Sentencia “if”:

Se trata de una estructura de control que permite redirigir un curso de una acción según la evaluación de una condición simple, sea falsa o verdadera.

Si la condición es verdadera, se ejecuta un bloque de código específico, o de lo contrario, se ejecutan las otras instrucciones del código según la sintaxis del código.

Se pueden plantear múltiples condiciones.

La sentencia “elif” va dentro de la sentencia “if”, siendo que “elif” es una condición dentro de la primera condición hasta que se cumpla alguna o ninguna siendo que si todas son falsas se ejecuta lo que está dentro del “else”.

Bucle “while”:

Es una estructura de control cuya función es meter una parte del código en un bucle hasta que la condición sea falsa (si esta llega a serlo), al contrario, terminar el programa o devolver el usuario al inicio para volver a ejecutar el código desde el inicio del código para evaluar la condición nuevamente.

Ejemplo de “while” e “if”:

```
while si <= 3:
    eleccion = input('Desea agregar un nuevo usuario(si/no): ')

    if eleccion == 'si':
        usuario=input('Ingrese el nuevo nombre de usuario: ')
        registro.append(usuario)

    elif eleccion == 'no':
        print('Muchas Gracias, vuelva pronto.')
        break
```

```
else:
    print('Usted no ha ingresado si o no.')

print(registro)
print('\n Muchas gracias, vuelva pronto \n')
```

sentencia “for”:

En general un bucle es una estructura de control que repite un determinado bloque de instrucciones un determinado número de veces.










El bucle de instrucciones que se repite se suele llamar cuerpo del bucle y cada repetición se suele llamar iteración.

Diagramas de flujo:

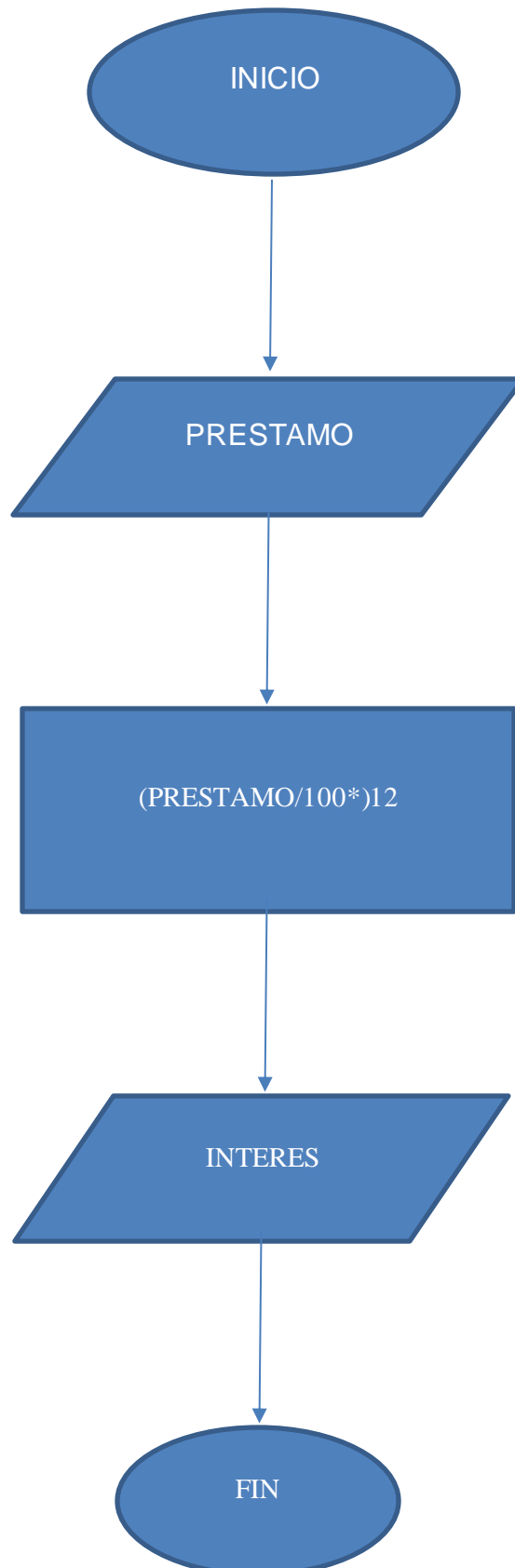
Es una manera de representar gráficamente un proceso o algoritmo, a través de una serie de pasos estructurados y vinculados que permiten su revisión como un todo, esta representación utiliza una determinada serie de figuras geométricas que definen la función que cumplen según su forma, estas determinan tal cual mapa una guía de inicio a fin de un programa.

Para que un diagrama de flujo tenga sentido necesitan un inicio y un final para terminar un proceso iniciado.

Los símbolos y significados de un diagrama de flujo:

Nombre	Símbolo	Función
Inicio/Final		Representa el inicio o fin de un proceso
Proceso		Representa la actividad llevada a cabo
Entrada/Salida		Representa información que ingresa o sale del sistema
Decisión		Indica un punto de toma de decisiones
Línea de Flujo		Indica el orden y sentido del flujo del proceso
Documento		Indica los documentos utilizados en el proceso
Base de datos		Representa la grabación de datos
Conector Interno/ Conector Externo		Enlace dentro de la misma página/ Enlace en diferente página
Retraso		Retraso para iniciar el siguiente proceso

Ejemplo de un diagrama de flujo:



Evidencia 2: Break, continue, listas e interacciones con las listas.

Break: El break es la palabra clave que fuerza a finalizar una estructura de control, en cambio el continue fuerza el inicio del siguiente ciclo de la estructura de control.

Listas: Las listas permiten guardar datos en una variable para acceder a ellos de la forma en que lo necesitemos, pueden ser numéricos o strings.

```
('\\n Registro 2021 \\n')
registro = []
si=1
while si <= 3:
    eleccion = input('Desea agregar un nuevo usuario(si/no): ')

    if eleccion == 'si':
        usuario=input('Ingrese el nuevo nombre de usuario: ')
        registro.append(usuario)

    elif eleccion == 'no':
        print('Muchas Gracias, vuelva pronto.')
        break

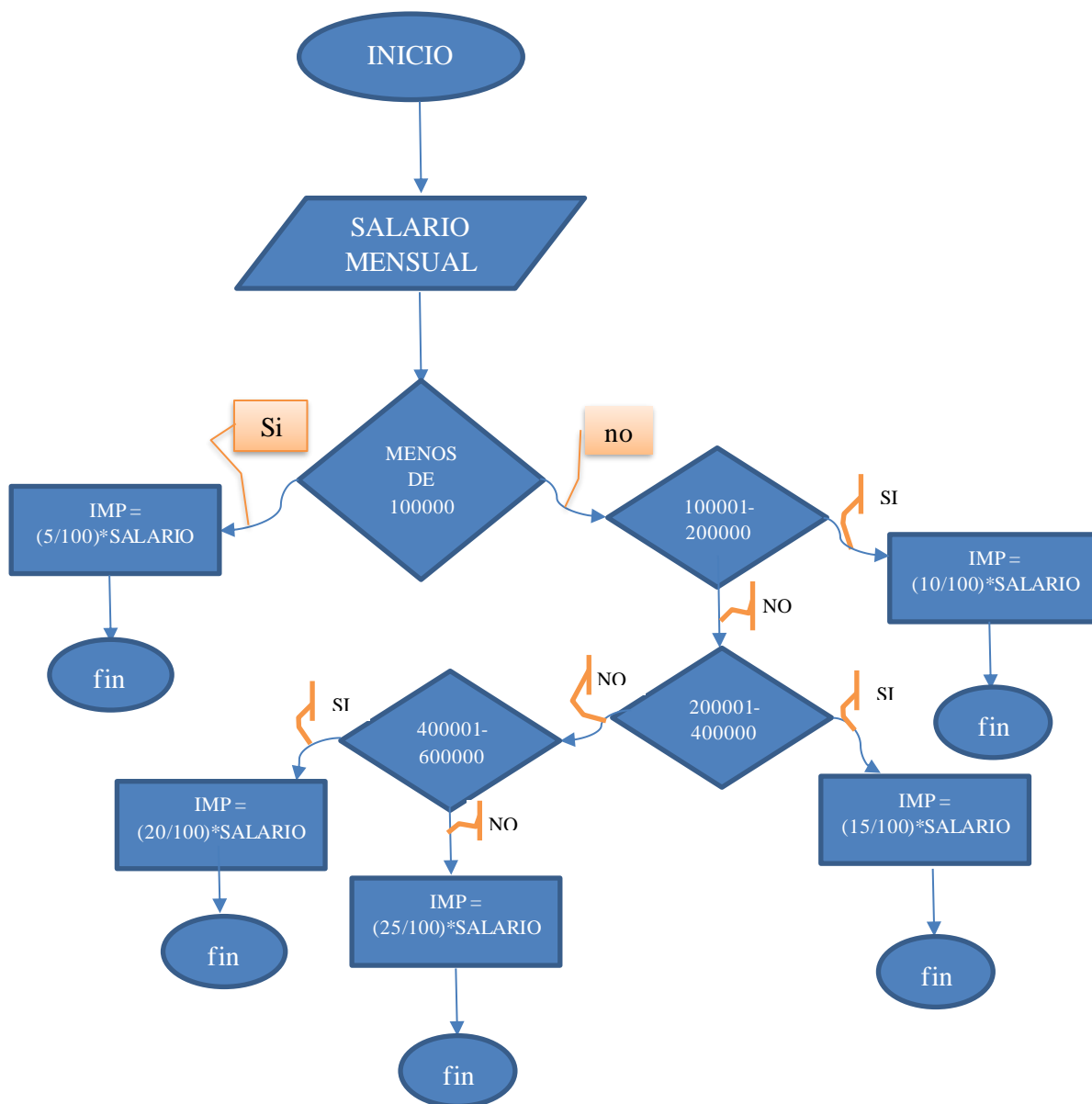
    else:
        print('Usted no ha ingresado si o no.')

print(registro)
print('\\n Muchas gracias, vuelva pronto \\n')
```

En este ejemplo se puede apreciar la función del break para detener la estructura de control while una vez que el usuario lo decida.

Así como se ve la función de las listas donde se irán agregando mas nombres hasta que el usuario decida acabar el programa y dejar de agregar nombres a la lista, al final del programa se muestra la lista de todos los nombres que ingreso el usuario.

Evidencia 3: Diagramas de flujo y código fuente (practica)



```
print("Bienvenido")

print("\nCalculador de impuestos mensuales\n")

salario_mensual = float(input("\nIngrese su salario mensual: \n"))

conidicion = 1

while conidicion == 1:
    dato = print("\nsu salario mensual es de: ",salario_mensual)

    if 0 < salario_mensual < 100000:
        imp1 = (5/100)*salario_mensual

        print('mensualmente usted pagara ',imp1,'de impuestos')
        break
    elif 100001 < salario_mensual < 200000:
        imp2 = (10/100)*salario_mensual

        print('mensualmente usted pagara ',imp2,'de impuestos')
        break
    elif 200001 < salario_mensual < 400000:
        imp3 = (15/100)*salario_mensual

        print('mensualmente usted pagara ',imp3,'de impuestos')
        break
    elif 400001 < salario_mensual == 600000:
        imp4 = (20/100)*salario_mensual

        print('mensualmente usted pagara ',imp4,'de impuestos')
        break
    else:
        imp5 = (25/100)*salario_mensual

        print('mensualmente usted pagara ',imp5,'de impuestos')
        break
```

Evidencia 4: Gestión de archivos

En base al Video de Google Drive “Gestión de archivos” se realizaron los siguientes programas en base a la gestión de archivos:

```
import io

files = open("datos.text","r")#r para leer los datos dentro del archivo
datos = files.readlines()
files.close()

print(datos)

nombre = input("Ingrese el nombre de la persona que desea buscar: ")

cuenta = datos.count(nombre+"\n")

if cuenta!=0:
    pos = datos.index(nombre+"\n")
    print("La edad de",nombre, "es de ",datos[(pos+1)])

else:
    print("no hay persona de nombre",nombre)
```

este es el programa que abre el archivo creado en el siguiente programa:

```
import io
print("datos de estudiantes")
nombre = input("\nDigite su nombre completo: ")
edad = input("\nDigite su edad en años: ")

files = open("datos.text","a")

files.write(nombre+"\n"+edad+"\n")

files.close()
```

Este programa guarda datos y los guarda en este caso en un bloc de notas, el programa anterior lee el bloc de notas donde se guardan los datos ingresados en el segundo programa.

Se puede apreciar gran utilidad a esta función de gestión de archivos para guardar múltiples datos que pueden ser de diversas utilidades en programas de compartimiento de información.