**GUÍA DE APRENDIZAJE Nº 5**

1. **IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Programa de Formación:  ANALISIS Y DESARROLLO DE SISTEMAS DE INFORMACION | Código:  Versión: | 228106  102 | | |
| Nombre del Proyecto:  IMPLEMENTACION DE UN APLICATIVO WEB PARA EL SEGUIMIENTO Y EVALUACION DE LA ETAPA PRODUCTIVA DE LOS APRENDICES SENA A TRAVES DE UNA BITACORA VIRTUAL | Código: | 1107755 | | |
| Fase del proyecto: | | ANALISIS | | |
| Actividad (es) del Proyecto:  Desarrollar el sistema | Actividad (es) de Aprendizaje:   * Crear módulos en python | Ambiente de formación ESCENARIO Aula 2030A | MATERIALES DE FORMACIÓN | |
| DEVOLUTIVO Televisor, tablero | CONSUMIBLE (unidades empleadas durante el programa) |
| **Resultados de Aprendizaje:**  Construir el modelo conceptual del macrosistema frente a los requerimientos del cliente, | Competencia:  Analizar los requerimientos del cliente para construir el sistema de información. | | | |
| Duración de la guía ( en horas): | 24 | | | |

1. **INTRODUCCIÓN**

|  |
| --- |
| Esta guía muestra cómo crear módulos en python. Los módulos, también llamados librerías, o APIS en otros lenguajes de programación, son un recurso muy importante a la hora de programar, ya que permiten por un lado, la reutilización de código, y por otra, organizar los grandes programas, separándolos y/o agrupándolos en paquetes; lo cual facilita el mantenimiento y/o posibles cambios en el código.  Python también cuenta con una librería estándar de módulos que facilita la resolución de problemas en diversas áreas como matemáticas, estadísticas, internet, etc. En están guía también se aprenderá como hacer uso de dicha biblioteca de una manera práctica; implementando pequeños programas que le permitirán apropiar este conocimiento. |

1. **ESTRUCTURACION DIDACTICA DE LAS ACTIVIDADES DE APRENDIZAJE**

|  |
| --- |
| * 1. **Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje.)**   En python cada archivo cuya extensión es .py, es sí mismo un módulo, el cual puede a su vez formar parte de un paquete; el paquete es una carpeta que contiene archivos . py. El paquete requiere tener dentro de sus archivos, uno denominado \_init\_.py , el cual puede ser un archivo vacío.    Los paquetes a su vez también pueden tener subpaquetes    Los módulos no necesariamente deben estar contenidos dentro de un subpaquete    **Importando módulos**  Para utilizar el contenido de un módulo, es necesario importarlo, lo cual debe hacerse con la instrucción **import** seguida del nombre del paquete más el nombre del módulo.  Ejemplos:  **import**  modulo # se importa un módulo que no pertenece a ningún paquete  **import**  paquete.modulo # se importa un módulo que está dentro de un paquete  **import**  paquete.subpaquete.modulo  Nota: Python tiene sus propios módulos, los cuales conforman la librería estándar.  **Namespaces**  Para hacer uso de cualquier elemento del módulo importado, se realiza mediante el namespace seguido de un punto (.) y el nombre del elemento que se quiera usar. El namespace es el nombre que se ha indicado luego de la palabra import; es decir la ruta del módulo.  Ejemplos:  **print** matematicas.PI#Se esta haciendo uso de la constante PI de modulo\_matematicas  **print** paquete.modulo\_matematicas.PI  **print** paquete.subpaquete.modulo\_matematicas.PI  **Alias**  Una manera de simplificar el uso de un módulo, es asignado un alias al namespace importado.  Ejemplo:  **import** modulo as m  **import** paquete.modulo as pm  **import** paquete.subpaquete.modulo as psm  Para hacer uso de cualquier elemento del módulo importado se haría así:  Ejemplo:  **print** m.PI#Se esta haciendo uso de la constante PI de modulo\_matematicas  **print** pm.PI  **print** psm.PI  **Importar elementos de un módulo sin importar namespaces**  También se pueden importar los elementos que se quieran de un módulo.  Ejemplo:  **from** paquete.modulo **import** PI  Así se puede hacer uso del elemento, directamente sin hacer uso del **namespace**.  **print** PI  También se puede importar más de un elemento así:  **from** paquete.modulo **import** CONSTANTE1, CONSTANTE2  Pero ¿qué sucede si los elementos importados desde módulos diferentes tienen los mismos nombres? En estos casos, habrá que prevenir fallos, utilizando alias para los elementos:  **from** paquete.modulo1 **import** CONSTANTE1 as C1, CONSTANTE2 as C2  **from** paquete.subpaquete.modulo1 **import** CONSTANTE1 as CS1, CONSTANTE2 as CS2  **print** C1  **print** C2  **print** C21  **print** CS1  También pueden importarse todos los elementos de un módulo de la siguiente manera:  **from** paquete.modulo **import** \*  Nota: de acuerdo al **PEP 8**,: la importación de módulos debe realizarse al comienzo del documento, en orden alfabético de paquetes y módulos. Primero deben importarse los módulos propios de Python. Luego, los módulos de terceros y finalmente, los módulos propios de la aplicación. Entre cada bloque de imports, debe dejarse una línea en blanco. |
| * 1. **Actividades de transferencia del conocimiento (Conceptualización y Teorización).**   **Importando módulos de la biblioteca estándar**  El siguiente programa importa el módulo **random,** y luego llama la función randint(), para generar números aleatorios:  import random  def juego():  dado1 = random.randint(1,6)  dado2 = random.randint(1,6)  print(f"Primer dado: {dado1}")  print(f"Segundo dado: {dado2}")  suma = dado1 + dado2  if (suma >= 7):  print ("Gano")  else:  print ("Perdio")  juego()  El siguiente programa hace uso de algunas funciones del módulo **math**  import math  def operaciones():  numero = int(input('Numero: '))  coseno = math.cos(numero)  tangente = math.tan(numero)  print(f'Coseno de {numero} es {coseno}')  print(f'Tangente de {numero} es {tangente}')  operaciones()  A continuación, se muestra el mismo programa anterior, pero haciendo uso de alias  import math as m  def operaciones():  numero = int(input('Numero: '))  print(f'Coseno de {numero} es {m.cos(numero)}')  print(f'Tangente de {numero} es {m.tan(numero)}')  operaciones()    El siguiente código muestra como importar componentes específicos del módulo **math**  from math import factorial,cos  def operaciones():  facto = factorial(5)  coseno = cos(5)  print (f"Factorial de 5 es {facto}")  print (f"Coseno de 5 es {coseno}")  operaciones()  **Creando módulos propios**  A continuación, se crea un módulo que será importando y utilizado por un segundo programa  **Modulo: matemáticas.py**  def factorial(numero):  facto = 1  for i in range(2,numero+1):  facto \*= i  return facto  A continuación, el programa que hace uso del módulo matemáticas  import matematicas as m  def factorial ():  numero = int(input("Numero: "))  print(f"Factorial de {numero} es {m.factorial(numero)}")  factorial()  **Actividad No.1**  Implemente un programa que ofrezca un menú con las siguientes opciones:   |  |  |  |  | | --- | --- | --- | --- | | PROGRAMA PYTHON UTILIZANDO MODULOS, INSTRUCCIONES REPETITIVAS Y CONDICIONALES | | | | |  |  |  |  | |  | 1. Factorial |  |  | |  | 1. Potencia de un numero | |  | |  | 1. Media | |  | |  | 1. Fibonacci 2. Salir |  |  | |  |  |  |  | |  |  |  |  | |  | Seleccione una opción [ ] | | |   Debe crear un módulo donde implemente las respectivas funciones, en el programa principal debe implementar las instrucciones necesarias para ejecutar iterativamente el menú con cada una de sus operaciones.  C:\Users\usuario\Google Drive\Luis Fernando Corredor\ADSI\Portafolio\Modulos\Implementacion\Curso Python\MisProgramas\matematicas.py  C:\Users\usuario\Google Drive\Luis Fernando Corredor\ADSI\Portafolio\Modulos\Implementacion\Curso Python\MisProgramas\menu\_matematicas.py |
| * 1. **Actividades de Apropiación del conocimiento (Conceptualización y Teorización).**       + Cree un módulo que contenga:   Una función que determine si un número es primo o no   * + - * Una función de retorne la potencia de un numero     - Implemente un programa que muestre un menú que con las siguientes opciones:       * Determinar si un número es primo o no       * Calcular la potencia de un número.   El programa debe hacer uso del módulo que creo. |
|  |

|  |
| --- |
|  |
|  |
| * 1. **Actividades de evaluación.**  |  |  |  | | --- | --- | --- | | **Evidencias de Aprendizaje** | **Criterios de Evaluación** | **Técnicas e Instrumentos de Evaluación** | | **Evidencias de Conocimiento :**   * Respuesta a preguntas.   **Evidencias de Desempeño:**   * Desarrollo de programas Java   **Evidencias de Producto:**   * Programas java | * Identifica y describe, en un sistema de información dado, los datos de entrada, * procesamiento de los datos e información generada, según necesidades del cliente. | * Pruebas de conocimiento * Lista de chequeo | |

1. **RECURSOS PARA EL APRENDIZAJE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ACTIVIDADES DEL PROYECTO** | **DURACIÓN  (Horas)** | **Materiales de formación devolutivos: (Equipos/Herramientas)** | | **Materiales de formación (consumibles)** | | **Talento Humano (Instructores)** | | **AMBIENTES DE  APRENDIZAJE TIPIFICADOS** |
| **Descripción** | **Cantidad** | **Descripción** | **Cantidad** | **Especialidad** | **Cantidad** | **ESCENARIO (Aula, Laboratorio, taller, unidad productiva)** y elementos y condiciones de seguridad industrial, salud ocupacional y medio ambiente |
|  |  |  |  |  |  |  |  |  |

|  |
| --- |
| **JDK**  **J2SE**  **Java** |

1. **GLOSARIO DE TERMINOS**
2. **REFERENTES BIBLIOGRÁFICOS**

|  |
| --- |
| Harvey M. Deitel, Paul J. Deitel (2004). Como Programar en Java, Pearson Education.  [Froufe Quintas, Agustín](http://biblioteca.sena.edu.co/F/8IVEUS4EGGBXJ5SGUXR2QRQLL3TFQ63B6GFACX6M9J3XX3KCS5-24793?func=full-set-set&set_number=009595&set_entry=000006&format=999) (2009). [Java 2 : manual de usuario y tutorial](http://biblioteca.sena.edu.co/F/8IVEUS4EGGBXJ5SGUXR2QRQLL3TFQ63B6GFACX6M9J3XX3KCS5-24794?func=full-set-set&set_number=009595&set_entry=000006&format=999), Alfaomega Ra-Ma  Flórez Fernández, Héctor Arturo (2012). Programación Orientada a Objetos Usando Java,  [Ecoe Ediciones](http://www.digitaliapublishing.com.bdigital.sena.edu.co/s/e/179).  <http://www.digitaliapublishing.com.bdigital.sena.edu.co/a/30030/programacion-orientada-a-objetos-con-java> |

1. **CONTROL DEL DOCUMENTO (ELABORADA POR)**

|  |
| --- |
| **Ing. Luis Fernando Corredor Mora** |