ILERNA Online

UF2: SISTEMAS ERP-CRM. EXPLOTACIÓN Y ADECUACIÓN

Tema 3. Desarrollo de componentes

Módulo MP10. Sistemas de gestión empresarial

PAC DESARROLLO RÚBRICA

PAC de Desarrollo UF1. Rúbrica

- **Explicación de la rúbrica**
  - **Criterios**
    - **A) Excelente:** La explicación es correcta, breve y clara.
    - **B) Bien:** Explicación bastante correcta, demasiado extensa y poco clara
    - **C) Regular:** La explicación esta incompleta o se ha copiado de los apuntes o es muy poco clara
    - **D) Mal:** Explicación incorrecta, No se ha realizado correctamente o se ha hecho otra cosa diferente a lo pedido

PAC DESARROLLO RÚBRICA

Criterio	A	B	C	D	TOTAL
1) Ventajas y Desventajas de ERP • Explicación correcta	0,25	0,15	0,1	0	0,25
2) Diferencias lógicas operacional y lógica analítica • Explicación de las diferencias y ejemplos	0,5	0,3	0,15	0	0,5
3) Explicación datawarehouse y sus funciones • Explicación correcta y las funciones	0,5	0,3	0,15	0	0,5
4) Información sobre Odoo • Se describe bien el ERP de Odoo	1	0,3	0,15	0	1
5) Comparativa ERPs • Se compara correctamente 3 ERPs	0,5	0,3	0,1	0	0,5
6) Investigación módulos Odoo • Se definen correctamente 3 módulos	1	0,7	0,3	0	1
7) Ejemplos módulos Odoo • Se explica el proceso y los pantallazos	1	0,7	0,3	0	1
8) Presentación del documento • Respuestas claras, buena presentación, sin faltas de ortografía, se pone bibliografía	0,25	0,15	0,1	0	0,25
					5

ILERNA Online

PAC Desarrollo – Revisión

Mensaje de revisión PAC Desarrollo

Solicito la revisión de la PAC de desarrollo con nota**(x).** Después de revisar la corrección con la rúbrica. La nota que me corresponde ha de ser de **(x).**

Expongo las preguntas y motivos de la revisión a continuación:

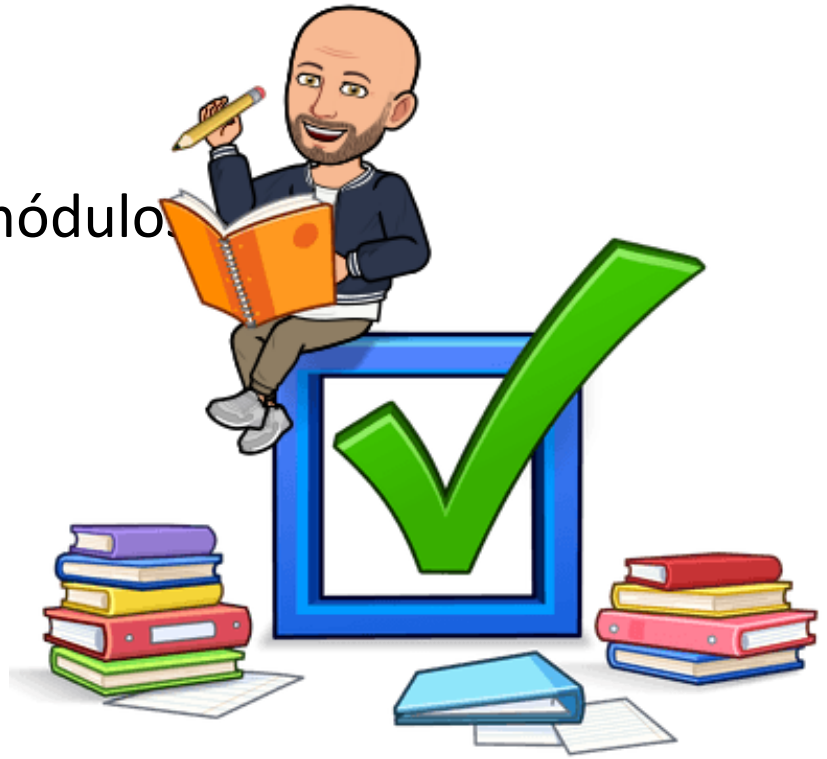
- Pregunta**(x)** - Nota actual**(x)** – Nota esperada tras la revisión **(x).** Los motivos son: (**explicación**)
- …

ILERNA Online

RESUMEN SESIÓN ANTERIOR

Conceptos vistos en la sesión anterior

- **Desarrollo de componentes**
  - Paso previo al desarrollo y programación de módulos
  - Lenguaje de programación Python
  - Las características de Python
  - Tipos de variables en Python
  - Impresión por pantalla
  - Variables



ERP-CRM. EXPLOTACIÓN Y ADECUACIÓN

Entornos de desarrollo

- **Instalación de Python**
  - **Eclipse**
  - **Pycharm**
    - PyCharm es un entorno de desarrollo integrado que se utiliza en programación informática, específicamente para el lenguaje Python.
  - **Interprete de código online:** https://repl.it/languages/p
  - **Crear un proyecto**
    - print “Hola Mundo”

ILERNA Online

ERP-CRM. EXPLOTACIÓN Y ADECUACIÓN

Repaso de conceptos de Python

- **Entrada/salida de datos - Variables - Tipos de datos**
  - #Python distingue mayúsculas y minúsculas.
    - Variable=5
    - variable=3
    - VaRiAbLe=8
  - #La variable “numero” tomará el último valor asignado. Valdrá 18 en es te caso
    - numero =5
    - numero =18
  - #Operacion sencilla
    - multiplicar = numero * variable
    - print(multiplicar)

ERP-CRM. EXPLOTACIÓN Y ADECUACIÓN

Repaso de conceptos de Python

- **Entrada/salida de datos - Variables - Tipos de datos**
  - #Programa que solicite al usuario que ingrese su nombre. El nombre se debe almacenar en una variable llamada nombre. Mostrar en pantalla el texto. “El nombre es…(nombre)”
    - nombre=input("Tu nombre:")
    - print("El nombre es: ", nombre)
  - #Programa que solicite al usuario ingresar la cantidad de kilómetros recorridos por una motocicleta y la litros de combustible que consumió durante ese recorrido. Mostrar el consumo por kilómetro.
    - kilometros=float(input("Kilómetros recorridos:"))
    - litros=float(input("Litros de combustible gastados:"))
    - print("El consumo por kilómetro es de", kilometros/litros)

ERP-CRM. EXPLOTACIÓN Y ADECUACIÓN

Características y sintaxis del lenguaje.

- **Control de flujo**
  - No se dispone de ningún elemento para poder indicar la finalización de un bloque, por lo que se establece el contenido del mismo mediante el sangrado, indicando el **mismo número de espacios a todas las líneas que formen parte de é**l(condicional/bucle).
  - De esta forma, existen distintos niveles de sangrado dependiendo de la cantidad de bloques existentes:
    - **Condicionales**: Mediante la cláusula if, se pueden realizar un conjunto de sentencias en función de una determinada condición.
    - **Bucles:**
      - **FOR**: Cuando hay que recorrer un objeto repetitivo (listas, tuplas, etc.) se puede hacer uso de la estructura for
      - **WHILE**: Python solo puede utilizar el bucle while <condición>.

Características y sintaxis del lenguaje.

- **Condicionales:**
  - Su construcción y funcionamiento con la clausula (if)
    - Mediante la cláusula if, se pueden realizar un conjunto de sentencias en función de una determinada condición.
      - La variante más simple que se presenta no cuenta con la parte del else y ejecuta todas las líneas que cumplan una condición.
      - Otra de las formas en las que puede aparecer es con las dos partes (if-else), ejecutando la parte del if en caso de que la condición sea verdadera y, para los demás casos, ejecutará la parte del else.
      - Otro caso que se debe señalar es que este lenguaje no cuenta con la tradicional sentencia switch o case. Por tanto, incorpora una estructura:
        - elif <condicion>
        - Se puede añadir a la condición if y else diferenciando los casos que sean necesarios.

```
valor = int(input("valor"))
print ("Menor igual a 10" if (valor<10) else "Mayor igual a 10")
```

Características y sintaxis del lenguaje.

- **Bucle for**
  - Cuando hay que recorrer un objeto repetitivo (listas, tuplas, etc.) se puede hacer uso de la estructura for, cuya sintaxis es la siguiente:
    - **for** varInicio **in** Objrepetitivo
    - El bucle debe recorrer cada uno de los componentes que pertenezcan a la lista, tomando un valor correspondiente en cada iteración.
- **Bucle while**
  - Python solo puede utilizar el bucle con un condicional
  - **while <condición>**
  - **Cuidado bucles infinitos**(comprobar que la condición se cumpla o que tengamos un break) y produciría un timeout, un error de sistema y se podría parar el sistema

Ejemplos Python

- **Ejemplos de condicionales**
  - Programa que solicite al usuario una letra y, si es una vocal, muestre el mensaje "Es vocal".
  - Verificar si el usuario ingresó un string de más de un carácter y, en ese caso,

```
letra=input("Letra:")
if len(letra)!=1:
    print("Debe ser sólo una letra")
else:
    if letra=="a" or letra=="e" or letra=="i" or letra=="o" or letra=="u" or\
            letra=="A" or letra=="E" or letra=="I" or letra=="O" or letra=="U":
        print("Es vocal")
    else:
        print("Es consonante")
```

Ejemplos Python

- **Ejemplos de bucle for**
  - Programa que permita al usuario ingresar 6 números enteros, que pueden ser positivos o negativos.
  - Al finalizar, mostrar la sumatoria de los números negativos y el promedio de los positivos.
  - No olvides que no es posible dividir por cero, por lo que es necesario evitar que [illegible] si no se ingresaron números positivos.

```
Número: 6
Número: 1
Número: -3
Número: -6
Número: 8
Sumatoria de los negativos:  -9
Promedio de los positivos:  5.0
```

Ejemplos Python

- **Ejemplos de bucle While**
  - Programa que, dada una frase por el usuario, la muestre invertida

```
frase=input("Frase:")
nueva=""
i=len(frase)-1
while i>=0:
    nueva=nueva+frase[i]
    i=i-1
print(nueva)
```

```
Frase:aloh
hola
```

Entornos de desarrollo

- **Programación Orientada a objetos**
  - En la Programación Orientada a Objetos debemos claro la diferencia entre una clase y un objeto
    - Una **clase** es una plantilla para la creación de objetos de datos según un modelo definido previamente. Las clases se utilizan para la definición de atributos (variables) y métodos (funciones).
    - Un **objeto** sería una instancia de esa clase, es decir, un objeto sería la llamada a una clase.

Ejemplos Python

- **Clases y objetos**
  - Realizar un programa que conste de una clase llamada Alumno que tenga como atributos el nombre y la nota del alumno. Definir los métodos para inicializar sus atributos, imprimirlos y mostrar un mensaje con el resultado de la nota y si h

```python
# inicializamos la clase
class Alumno:
    # inicializamos los atributos
    def inicializar(self, nombre, nota):
        self.nombre = nombre
        self.nota = nota

    # función para imprimir los datos
    def imprimir(self):
        print("Nombre: ", self.nombre)
        print("Nota: ", self.nota)

    # función para obtener el resultado
    def resultado(self):
        if self.nota < 5:
            print("El alumno ha suspendido")
        else:
            print("El alumno ha aprobado")
```

Entornos de desarrollo

- **Programación Orientada a objetos**
  - **Herencia en Python**
    - En Python dos clases además de poder tener una relación de colaboración, también pueden tener una relación de herencia.
    - La herencia significa que se pueden crear nuevas clases partiendo de otras clases ya existentes, que heredarán todos los atributos y métodos de su clase padre además, de poder añadir los suyos propios.
    - Por ejemplo, si tenemos una clase llamada vehículo, esta sería la clase padre de las clases coche, moto, bicicleta… Cada una de estas subclases tendría los atributos y métodos de su padre vehículo y aparte tendrían sus propios métodos cada uno de ellos.

ERP-CRM. EXPLOTACIÓN Y ADECUACIÓN

Ejemplos Python

```python
# declaramos la clase empleado
# la clase empleado hereda los atributos y metodos de la clase Persona
class Empleado(Persona):
    # declaramos el metodo __init__
    def __init__(self):
        # llamamos al metodo init de la clase padre
        # utilizamos la funcion super() para hacer referencia al padre
        super().__init__()
        self.sueldo = float(input("Ingrese el sueldo: "))
```

- **Herencia**
  - Realizar un programa que conste de un clase Persona con dos atributos nombre y edad. Los atributos se introducirán por teclado y habrá otro método para imprimir los datos. Declarar una segunda clase llama Empleado que hereda de la clase Persona y agrega el atributo sueldo. Debe mostrar si tiene que pagar impuestos o no (sueldo superior a 3000).
    - La clase Persona no tiene nada nuevo y creamos un objeto de la misma forma que lo realizado hasta ahora. Por el contrario en la clase Empleado sí que vemos cosas nuevas.
    - Añadiendo entre paréntesis la clase Persona, estamos especificándole que hereda dicha clase. Entonces podemos decir que la clase Empleado heredará los atributos de Persona.
    - Otra cosa nueva es la función super(), con la cual hacemos referencia a la clase heredada.

Ejemplos Python

- **Enunciado**
  - Función que reciba un string y retorne True si es un palíndromo (esto es, si se lee igual de izquierda a derecha o de derecha a izquierda), False en caso contrario. Utilizar esta función en un programa que permita al usuario ingresar palabras hasta que ingrese la palabra “fin” (suponer que todas las palabras son en minúsculas o todas en mayúsculas, de forma consistente).
  - Al finalizar, mostrar la cantidad de palíndromos ingresados.
    - Cadena: abba
    - Cadena: m
    - Cadena: luz
    - Cadena: reconocer
    - Cadena: golondrina
    - Cantidad de palíndromos: **3**

Ejemplos Python

- **Enunciado**
  - Realizar un programa en el cual se declaren dos valores enteros por teclado utilizando el método __init__.
  - Calcular después la suma, resta, multiplicación y división.
  - Utilizar un método para cada una e imprimir los resultados obtenidos.
  - Llamar a la clase Calculadora.

Entornos de desarrollo

- **Módulos y Librerías en Python : Importar, acceder, crear**
  - A la hora de programar a veces es necesario recurrir a módulos, librerías, paquetes, etc para facilitarnos el desarrollo de un programa sin tener que repetir código o inventar la rueda nuevamente. También para organizar nuestro programa si es demasiado extenso y cuenta con muchas líneas de código, su estructura puede ser separada en módulos.
  - **Módulo en programación**
    - En programación un módulo se define como la porción de un programa.
  - **Librería en programación (biblioteca)**
    - En programación una librería se define como un conjunto de implementaciones funcionales.

Entornos de desarrollo

- **Importando módulos y librerias**
  - Bien si un módulo es como una caja de herramientas y nosotros vamos a hacer cálculos, necesitamos un módulo que posea estos accesorios y debemos incorporarlo a nuestro programa.
  - Vamos a crear un archivo .py como clásicamente lo hacemos en nuestro IDE. Y dentro de él vamos a importar el módulo Math usando la orden import. De

```
#Importamos el módulo math
import math
x = int(input("Ingresa un numero \n"))
#Utilizamos la función sqrt del módulo math
raiz = math.sqrt(x)
print (raiz)
```

Ejemplos Python

- **Enunciado**
  - En un banco tienen clientes que pueden hacer depósitos y extracciones de dinero.
  - El banco requiere también al final del día calcular la cantidad de dinero que se ha depositado.
  - Se deberán crear dos clases, la clase cliente y la clase banco. La clase cliente tendrá los atributos nombre y cantidad y los métodos __init__, depositar, extraer, mostrar_total.
  - La clase banco tendrá como atributos 3 objetos de la clase cliente y los métodos __init__, operar y deposito_total.

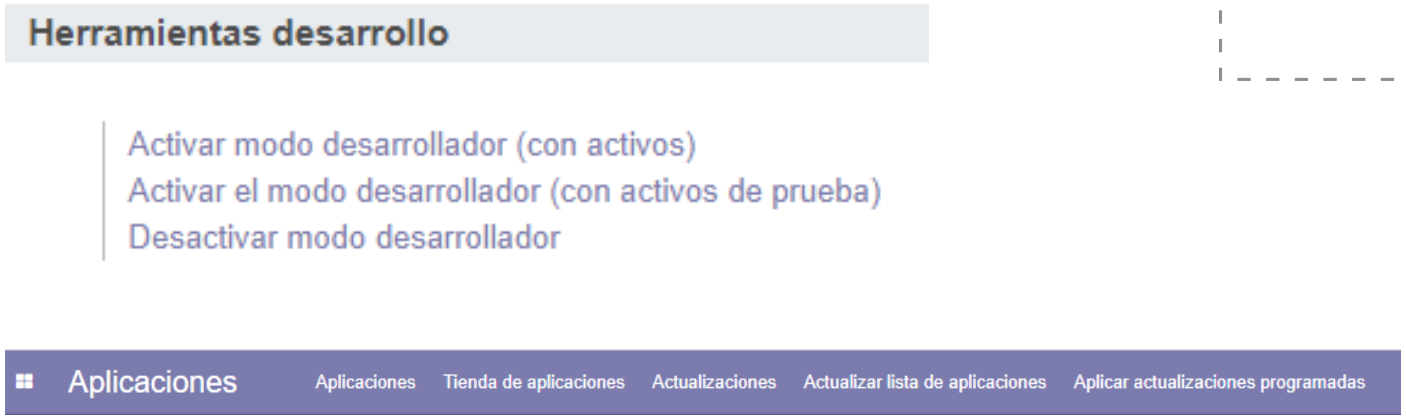
¿DUDAS?

ILERNA Online

SISTEMAS ERP-CRM. IMPLANTACIÓN

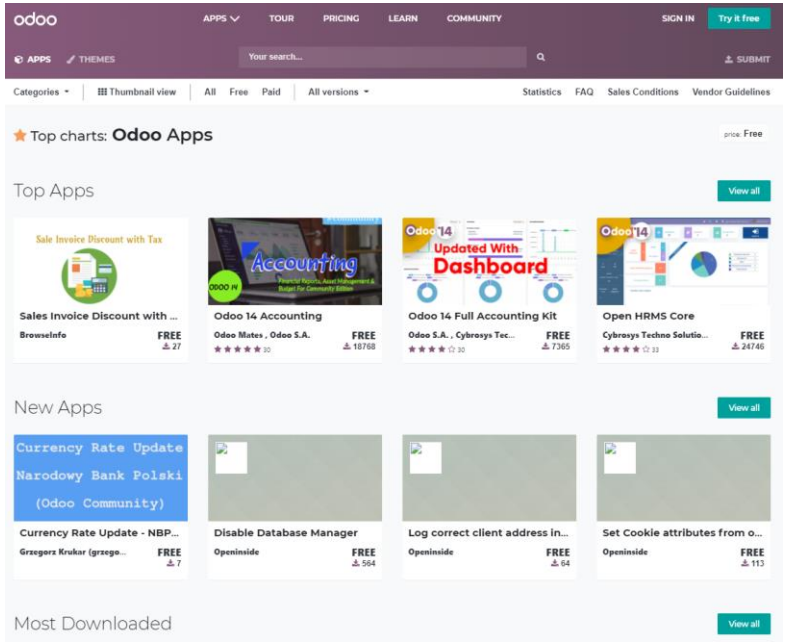
Administración básica y configuración

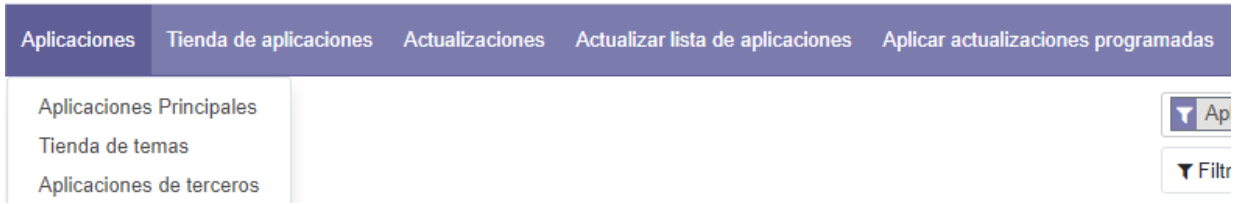
- **Activar Modo desarrollador**
  - Para tener más opciones de configuración.
  - Activar modo desarrollador en Ajustes Generales



Administración básica y configuración

- **Añadir Apps de terceros**
  - Bajar App
  - Descomprimir en el archivo de addons
    - C:\Program Files\Odoo 14.0.20201021\server\odoo\addons
  - Actualizar lista de Odoo
  - Instalar App
  - Lista para usarse





¿DUDAS?

ILERNA Online