Quiz 1

Fecha de entrega 10 de abr en 13:00 **Puntos** 20 **Preguntas** 3
Disponible 10 de abr en 11:45 - 10 de abr en 13:00 1 hora y 15 minutos
Límite de tiempo 30 minutos

Instrucciones

Lea con detenimiento cada una de las preguntas que se le presentan y responda de acuerdo a lo que se le solicita.

Este examen fue bloqueado en 10 de abr en 13:00.

Historial de intentos

	Intento	Hora	Puntaje
MÁS RECIENTE	Intento 1	22 minutos	20 de 20

Puntaje para este examen: **20** de 20
Entregado el 10 de abr en 12:10
Este intento tuvo una duración de 22 minutos.

Pregunta 1 — 6 / 6 pts

Crea una clase base llamada Animal con atributos name y sound. Luego, crea subclases para tres (3) animales específicos (por ejemplo, Perro, Gato, Loro). Cada subclase debe sobrescribir el atributo sound y proporcionar un sonido específico para ese animal. Copie el código con su respuesta en el lugar indiccado

Su respuesta:

```
class Animal:
    def __init__(self, name, sound):
        self.name = name
        self.sound = sound

    def animal_sound(self):
        print(f"El {self.name} hace {self.sound}")
```

```
class Oso(Animal):
    def __init__(self, name):
        super().__init__(name, "Roaar")

class Gato(Animal):
    def __init__(self, name):
        super().__init__(name, "miau")

class Leon(Animal):
    def __init__(self, name):
        super().__init__(name, "Grrr")


oso = Oso("Oso")
gato = Gato("Gato")
leon = Leon("Leon")

oso.animal_sound()
gato.animal_sound()
leon.animal_sound()
```

Pregunta 2

7 / 7 pts

En la página oficial de **Python (https://www.python.org/)** , en el carrusel de la página principal, existe un código para generar la serie de Fibonacci de forma iterativa con un ciclo while. Modifique la funcionalidad para que sea realizada con una función generadora, luego con un ciclo iterativo usando la función realizada entregue los primeros 15 números de la serie. Copia tu código con la respuesta en el lugar indicado.

Su respuesta:

```
def fibonacci_funct_generadora():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b
```

```
fibonacci = fibonacci_funct_generadora()

for _ in range(7):
    print(next(fibonacci), end=' ')
print()
```

Pregunta 3

7 / 7 pts

La palabra **with** se usa en Python para gestionar un contexto, es así como se puede abrir un archivo usando with y no hay necesidad de cerrarlo, ya que el gestor se encarga de ello. Para crear un gestor de contexto se implemetan los métodos __enter__() y __exit__() para definir la entrada y salida respectivamente del contexto. Cree un clase llamada Timer, que aplicada a una iteración cualquiera mida el tiempo de ejecución de la iteración.

Ayuda: debe importar el módulo time y usar el método time() para obtener el tiempo actual de la máquina. Además en la implemnetación del metodo __exit__() debe tener un mensaje que indique el tiempo transcurrido en segundos. Para probar la funcionalidad de su clase hágalo con el siguiente código:

```
with Timer():
    # Bloque de código para medir el tiempo de ejecución
    for _ in range(1000000):
        pass
```

Copia tu código con la respuesta en el lugar indicado.

Su respuesta:

```
import time

class Timer:
    def __enter__(self):
        self.inicio_time = time.time()

    def __exit__(self, exc_type, exc_val, exc_tb):
        fin_time = time.time()
        transcurrido_time = fin_time - self.inicio_time
```

```
        print(f"El tiempo transcurrido es: {transcurrido_time} segundos")

with Timer():
    for _ in range(1000000):
        pass
```

Puntaje del examen: **20** de 20

Quiz 1

Fecha de entrega 10 de abr en 13:00 **Puntos** 20 **Preguntas** 3
Disponible 10 de abr en 11:45 - 10 de abr en 13:00 1 hora y 15 minutos
Límite de tiempo 30 minutos

Detalles de la entrega:

Hora:	29 minutos
Puntaje actual:	19 de 20
se mantuvo el puntaje:	19 de 20

Instrucciones

Lea con detenimiento cada una de las preguntas que se le presentan y responda de acuerdo a lo que se le solicita.

Este examen fue bloqueado en 10 de abr en 13:00.

Historial de intentos

	Intento	Hora	Puntaje
MÁS RECIENTE	Intento 1	29 minutos	19 de 20

Puntaje para este examen: **19** de 20
Entregado el 10 de abr en 12:15
Este intento tuvo una duración de 29 minutos.

Pregunta 1

5 / 6 pts

Crea una clase base llamada Animal con atributos name y sound. Luego, crea subclases para tres (3) animales específicos (por ejemplo, Perro, Gato, Loro). Cada subclase debe sobrescribir el atributo sound y proporcionar un sonido específico para ese animal. Copie el código con su respuesta en el lugar indiccado

Su respuesta:

```
from abc import ABC, abstractmethod
#clase abstracta
class Animal(ABC):
    name:str
    sound:str

    @abstractmethod
    def make_sound(self):
        pass
#clases concreta
class Loro(Animal):
    def __init__(self):
        super().__init__('Loro')
        self.sound = 'romeo'
        self.name = 'loro'

    def make_sound(self):
        return self.sound
 #clases concreta
class Perro(Animal):
    def __init__(self):
        super().__init__('Perro')
        self.sound = 'guau'
        self.name = 'Perro'

    def make_sound(self):
        return self.sound

#clases concreta
class Gato(Animal):
    def __init__(self):
        super().__init__('Gato')
        self.sound = 'Miau'
        self.name = 'Gato'

    def make_sound(self):
        return self.sound
```

No había que hacer una clase abstracta

Pregunta 2

7 / 7 pts

En la página oficial de Python, en el carrusel de la página principal, existe un código para generar la serie de Fibonacci de forma iterativa con un ciclo while. Modifique la funcionalidad para que sea realizada con una función generadora, luego con un ciclo iterativo usando la función realizada entregue los primeros 15 números de la serie. Copia tu código con la respuesta en el lugar indicado.

Su respuesta:
#pagina de python:

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
```

#funcion generadora:

```
def fibonacci():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

fibonacciR = (x for x in fibonacci())
for i, number in enumerate(fibonacciR, 1):
    print(number)
    if i == 15:
        break
```

Pregunta 3

7 / 7 pts

La palabra **with** se usa en Python para gestionar un contexto, es así como se puede abrir un archivo usando with y no hay necesidad de cerrarlo, ya que el gestor se encarga de ello. Para crear un gestor de contexto se implemetan los métodos __enter__() y __exit__() para definir la entrada y salida respectivamente del contexto. Cree un clase llamada Timer, que aplicada a una iteración cualquiera mida el tiempo de ejecución de la iteración.

Ayuda: debe importar el módulo time y usar el método time() para obtener el tiempo actual de la máquina. Además en la implemenetación del metodo __exit__() debe tener un mensaje que indique el tiempo transcurrido en segundos. Para probar la funcionalidad de su clase hágalo con el siguiente código:

```
with Timer():
    # Bloque de código para medir el tiempo de ejecución
    for _ in range(1000000):
        pass
```

Copia tu código con la respuesta en el lugar indicado.

Su respuesta:

```
import time

class Timer:
    def __enter__(self):
        self.start_time = time.time()
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        end_time = time.time()
        elapsed_time = end_time - self.start_time
        print(f"Tiempo transcurrido: {elapsed_time:.6f} segundos")

# Prueba de la clase Timer
with Timer():
    for _ in range(1000000):
        pass
```

Puntaje del examen: **19** de 20

Quiz 1

Fecha de entrega 2 de oct de 2024 en 13:00 **Puntos** 20 **Preguntas** 4
Disponible 2 de oct de 2024 en 12:00 - 2 de oct de 2024 en 13:00 1 hora
Límite de tiempo 40 minutos

Instrucciones

Lea con detenimiento cada una de las preguntas que se le presentan a continuación y responda de acuerdo a lo solicitado

Este examen fue bloqueado en 2 de oct de 2024 en 13:00.

Historial de intentos

	Intento	Hora	Puntaje
MÁS RECIENTE	Intento 1	30 minutos	18 de 20

Puntaje para este examen: **18** de 20
Entregado el 2 de oct de 2024 en 12:33
Este intento tuvo una duración de 30 minutos.

Pregunta 1 — 4 / 4 pts

Explique con sus propias palabras la diferencia entre una lista y una tupla en Python

Su respuesta:

Básicamente la diferencia principal radica que la lista es mutable, mientras que la tupla es inmutable. Esto implica que en tiempo de ejecución la listar puede ser modificada, ya sea agregando elementos, o modificando los valores dentro del mismo. Caso contrario con las tuplas.

In responder

Pregunta 2

2 / 4 pts

Se tiene una lista de números enteros, la cual contiene valores repetidos, se desea crear una lista con los números no repetidos de la primera lista. Para probar use la siguente lista:
13, 10, 22, 18, 32, 39, 21, 5, 33, 15, 18, 2, 9, 31, 18, 7, 23, 27, 28, 9, 31, 15, 7, 28, 18
Al final imprima ambas listas y la cantidad de elementos de cada una
Nota:
puede usar el operador “in”, pero no el concepto de “set”

Su respuesta:

Respondido en correo: Muy buen algoritmo, lastima que no completó el ejercicio

Pregunta 3

5 / 5 pts

Defina un diccionario llamado informacion_persona, que contenga información básica de una persona: nombre, edad, ciudad y correo electrónico.

En el mismo código cree una variable que acceda a la información de edad y la imprima en el terminal de la computadora.

Su respuesta:

```
informacion_persona = {'nombre': 'Paul','edad': 18,'ciudad':'Caracas', 'correo': 'correo@prueba.com'}
edad_persona = informacion_persona['edad']
print(f"La edad de la persona es: {edad_persona}")
```

In responder

Pregunta 4

7 / 7 pts

Cree una clase base llamada Employee con atributos name, employee_id y salary. Luego, cree subclases para diferentes tipos de empleados (por ejemplo, Gerente, Desarrollador, Vendedor). Cada subclase debe tener atributos y métodos adicionales específicos para su rol.

Recuerde sobreescribir el método __str__() para mostrar su representación.

A contininuación instancie objetos de tipo Gerente, Desarrollador y otro de tipo Vendedor, e imprima sus atributos por la pantalla.

Copie su código y péguelo en el lugar indicado. Revise que la indentación quedó bien configurada.

Su respuesta:

Enviada por correo: EN vez de escribir tanto, podía sobreescribir el método __str__() de la super clase y agregar la información adicional

Puntaje del examen: **18** de 20

Quiz 1

Fecha de entrega 2 de oct de 2024 en 13:00 **Puntos** 20
Preguntas 4
Disponible 2 de oct de 2024 en 12:00 - 2 de oct de 2024 en 13:00 1 hora
Límite de tiempo 40 minutos

Instrucciones

Lea con detenimiento cada una de las preguntas que se le presentan a continuación y responda de acuerdo a lo solicitado

Este examen fue bloqueado en 2 de oct de 2024 en 13:00.

Historial de intentos

	Intento	Hora	Puntaje
MÁS RECIENTE	Intento 1	40 minutos	14 de 20

Puntaje para este examen: **14** de 20
Entregado el 2 de oct de 2024 en 12:43
Este intento tuvo una duración de 40 minutos.

Pregunta 1 — 4 / 4 pts

Explique con sus propias palabras la diferencia entre una lista y una tupla en Python

Su respuesta:

Una lista tiene elementos que pueden ser modificables y es dinámica, mientras que la tupla una vez definida sus elementos no puede modificarse.

Pregunta 2

4 / 4 pts

Se tiene una lista de números enteros, la cual contiene valores repetidos, se desea crear una lista con los números no repetidos de la primera lista. Para probar use la siguente lista:
13, 10, 22, 18, 32, 39, 21, 5, 33, 15, 18, 2, 9, 31, 18, 7, 23, 27, 28, 9, 31, 15, 7, 28, 18
Al final imprima ambas listas y la cantidad de elementos de cada una
Nota:
puede usar el operador “in”, pero no el concepto de “set”

Su respuesta:

```
lista = [13, 10, 22, 18, 32, 39, 21, 5, 33, 15, 18, 2, 9, 31, 18, 7, 23, 27, 28, 9, 31, 15, 7, 28, 18]
lista_norepetida = []
for i in lista:
    if (i not in lista_norepetida):
        lista_norepetida.append(i)


print(f"La lista completa tiene una longitud {lista.__len__()} y sus elementos son:")
print(lista)
print("-----------------")
print(f"La lista sin elementos repetidos tiene una longitud {lista_norepetida.__len__()} y sus elementos son:")
print(lista_norepetida)
```

n responder

Pregunta 3

0 / 5 pts

Defina un diccionario llamado informacion_curso, que contenga información básica de un curso: nombre, descripción, duración y costo.

En el mismo código cree una variable que acceda a la información del nombre y la imprima en el terminal de la computadora.

Su respuesta:

Pregunta 4

6 / 7 pts

Cree una clase llamada Persona con los siguientes atributos: nombre, fecha de nacimiento (como un string) y cédula de identidad. Luego cree dos clases que herenden de Persona:

1. una llamada Estudiante, con un atributo adicional denominado carrera, para indicar que carrera cursa

2. otra llamada Profesor, con un atributo adicional denominado asignatura, para indicar la materia que dicta.

En ambos casos recuerde sobreescribir el método __str__() para mostrar su representación.

A contininuación instancie dos objetos uno de tipo Estudiante y otro de tipo Profesor, e imprima sus atributos por la pantalla.

Su respuesta:

```
class Estudiante(Persona):
    def __init__(self, nombre, f_nac, cedula, carrera):
        super().__init__(nombre, f_nac, cedula)
        self.carrera = carrera
    def __str__(self):
        return f"Soy {self.nombre}, un estudiante, y estudio la carrera {self.carrera}. Cedula: {self.cedula},  fecha nacimiento{self.f_nac}"


class Profesor(Persona):
    def __init__(self, nombre, f_nac, cedula, asignatura):
        super().__init__(nombre, f_nac, cedula)
        self.asignatura = asignatura

    def __str__(self):
        return f"Soy {self.nombre}, un profesor y doy la asignatura
```

```
{self.asignatura}. Cedula: {self.cedula},  fecha nacimiento
{self.f_nac}"



estudiante = Estudiante("Nadine", "29/08/99", 26838242,
"Ingenieria Informatica")
profesor = Profesor("Carlos", "29/08/72", 17158551, "Desarrollo de
Software")


print(estudiante)
print(profesor)
```

¿y donde está la clase Persona?

Puntaje del examen: **14** de 20

¡Correcto!

- [x] Modificar el comportamiento de una función sin modificar su código fuente.

Pregunta 2

1 / 1 pts

¿Cuál es la ventaja principal de usar funciones generadoras en lugar de listas estándar en Python?

- [ ] Las funciones generadoras son más rápidas en la ejecución.
- [ ] Las funciones generadoras pueden almacenar más datos que las listas.

¡Correcto!

- [x] Las funciones generadoras ocupan menos espacio en la memoria.
- [ ] Las funciones generadoras permiten modificar los valores almacenados.

Pregunta 3

1 / 1 pts

¿Cuál es el método dunder en Python utilizado para la representación de cadena de un objeto?

- [ ] __repl__
- [ ] __format__
- [ ] __string__

¡Correcto!

- [x] __str__

Parcial 1

Fecha de entrega 8 de mayo en 12:40 **Puntos** 20 **Preguntas** 8
Disponible 8 de mayo en 10:10 - 8 de mayo en 12:40 2 horas y 30 minutos
Límite de tiempo 150 minutos

Detalles de la entrega:

Hora:	62 minutos
Puntaje actual:	20 de 20
se mantuvo el puntaje:	20 de 20

Instrucciones

Lea con detenimiento cada una de. las preguntas que se le presentan y responda de acuerdo a lo solicitado.

Durante el examen no está permitido el uso de celulares.

Podrá hacer uso de las herraminetas de módulo 7, así como de documentación que requiera de. la web.

Este examen fue bloqueado en 8 de mayo en 12:40.

Historial de intentos

	Intento	Hora	Puntaje
MÁS RECIENTE	Intento 1	62 minutos	20 de 20

Puntaje para este examen: **20** de 20
Entregado el 8 de mayo en 11:16
Este intento tuvo una duración de 62 minutos.

Pregunta 1

1 / 1 pts

¿Cuál es la sintaxis básica para aplicar un decorador a una función en Python?

¡Correcto!

- (●) @decorador
- ( ) función @decorador()
- ( ) decorador:
- ( ) función decorador()

Pregunta 2

1 / 1 pts

¿Cuál es la ventaja principal de usar funciones generadoras en lugar de listas estándar en Python?

¡Correcto!

- (●) Las funciones generadoras ocupan menos espacio en la memoria.
- ( ) Las funciones generadoras pueden almacenar más datos que las listas.
- ( ) Las funciones generadoras permiten modificar los valores almacenados.
- ( ) Las funciones generadoras son más rápidas en la ejecución.

Pregunta 3

1 / 1 pts

¿Cuál es el método dunder en Python utilizado para la representación de cadena de un objeto?

- ( ) __repl__
- ( ) __string__
- (●) __str__ ¡Correcto!
- ( ) __format__

Pregunta 4

1 / 1 pts

¿Cuál es el propósito principal del comando **pip** en un ambiente virtual?

¡Correcto!

- (●) Instalar paquetes y bibliotecas de Python.
- ( ) Crear un nuevo ambiente virtual.
- ( ) Actualizar Python a la última versión.
- ( ) Desinstalar Python completamente.

Pregunta 5

1 / 1 pts

En un schema Pydantic, ¿cómo se puede forzar que un campo numérico, digamos `age`, sea mayor o igual a 18?

- ( ) `age: int = Field(..., min=18)`
- ( ) `age: int = Field(..., gt=18)`
- (●) `age: int = Field(..., ge=18)` ¡Correcto!
- ( ) `age: int = Field(..., min_value=18)`

Pregunta 6

4 / 4 pts

¿Cómo se implementa la función dunder __len__ para obtener la longitud de una estructura de datos personalizada, llámela MiEstructura? (Muestre el código)

Para probar use lo siguiente:

```
mi_estructura = MiEstructura([1, 2, 3, 4, 5])
print(len(mi_estructura))  # Debería imprimir 5
```

Su respuesta:

```
class MiEstructura:
    def __init__(self, datos):
        self.datos = datos

    def __len__(self):
        return len(self.datos)

mi_estructura = MiEstructura([1, 2, 3, 4, 5])
print(len(mi_estructura))
```

Pregunta 7

5 / 5 pts

Desarrolla una función generadora llamada pares_hasta(límite) que produzca números pares hasta un límite superior establecido.

para probar use lo siguiente:

```
pares_hasta_20 = pares_hasta(20)  # Generador de pares hasta 20

for par in pares_hasta_20:
    print(par)  # Salida: 2 4 6 8 10 12 14 16 18 20
```

Su respuesta:

```
def pares_hasta(límite):
    numero = 2
    while numero <= límite:
        yield numero
        numero += 2

pares_hasta_20 = pares_hasta(20)

for par in pares_hasta_20:
    print(par)
```

Pregunta 8

6 / 6 pts

Cree un aplicación simple con FastAPI que tenga cuatro endpoint tres GET y un POST. En este caso va a usar un schema Pydantic llamado Empleado, el cual va tener un id(int), una nombre completo(str), cargo(str), sueldo(str) y departamento (Enum.Departamento), este último es un Enumerado que tendrá cuatro valores str: Finanzas, RRHH, Producción y TRansporte. Los datos se guardaran en una lista que usará como base de datos.
El POST, incluirá un Empleado en la lista
El primer GET estará en la base de la aplicación e indicará un mensaje: "Bienvenidos a la aplicación de ejemplo"
El segundo GET mostrará todos los empleados, en el path /employees/
El segundo GET mostrará un empleado por su ID, en el path /employees/{id}

Su respuesta:

```
from fastapi import FastAPI, Body
from pydantic import BaseModel, Field
from enum import Enum

class Departamento(Enum):
    finanzas = "Finanzas"
    rrhh = "RRHH"
    produccion = "Producción"
    transporte = "Transporte"


class Empleado(BaseModel):
    id: int = Field(..., ge=1)
    nombre_completo: str = Field(...)
    cargo: str = Field(...)
    sueldo: str = Field(...)
    departamento: Departamento = Field(...)

empleados = []


app = FastAPI()


@app.get("/")
def bienvenida():
    return {"mensaje": "Bienvenidos a la aplicación de ejemplo"}


@app.get("/employees/")
def obtener_empleados():
    return empleados


@app.get("/employees/{id}")
def obtener_empleado_por_id(id: int):
    for empleado in empleados:
        if empleado.id == id:
            return empleado
    return {"mensaje": "Empleado no encontrado"}


@app.post("/employees/")
def crear_empleado(empleado: Empleado = Body(...)):
    empleados.append(empleado)
    return empleado
```

Puntaje del examen: **20** de 20

Parcial 1

Fecha de entrega 8 de mayo en 12:40 **Puntos** 20 **Preguntas** 8
Disponible 8 de mayo en 10:10 - 8 de mayo en 12:40 2 horas y 30 minutos
Límite de tiempo 150 minutos

Instrucciones

Lea con detenimiento cada una de. las preguntas que se le presentan y responda de acuerdo a lo solicitado.

Durante el examen no está permitido el uso de celulares.

Podrá hacer uso de las herramientas de módulo 7, así como de documentación que requiera de. la web.

Este examen fue bloqueado en 8 de mayo en 12:40.

Historial de intentos

	Intento	Hora	Puntaje
MÁS RECIENTE	Intento 1	48 minutos	19 de 20

Puntaje para este examen: **19** de 20
Entregado el 8 de mayo en 11:04
Este intento tuvo una duración de 48 minutos.

Pregunta 1 — 1 / 1 pts

¿Qué se puede hacer con un decorador en Python?

- Convertir una función en una clase.
- Cambiar el tipo de datos de una variable.
- Cambiar el nombre de una función.

¡Correcto!

- [x] Modificar el comportamiento de una función sin modificar su código fuente.

Pregunta 2

1 / 1 pts

¿Cuál es la ventaja principal de usar funciones generadoras en lugar de listas estándar en Python?

- [ ] Las funciones generadoras son más rápidas en la ejecución.
- [ ] Las funciones generadoras pueden almacenar más datos que las listas.

¡Correcto!

- [x] Las funciones generadoras ocupan menos espacio en la memoria.
- [ ] Las funciones generadoras permiten modificar los valores almacenados.

Pregunta 3

1 / 1 pts

¿Cuál es el método dunder en Python utilizado para la representación de cadena de un objeto?

- [ ] __repl__
- [ ] __format__
- [ ] __string__

¡Correcto!

- [x] __str__

Pregunta 4

1 / 1 pts

¿Qué comando se utiliza para activar un ambiente virtual en Python?

¡Correcto!

- (●) En Windows: nombre_del_ambiente\Scripts\activate, En macOS y Linux: source nombre_del_ambiente/bin/activate'
- ( ) En todos los SO: activate nombre_del_ambiente
- ( ) En Windows: start-environment nombre_del_ambiente, En macOS y Linux: start-env nombre_del_ambiente
- ( ) En Windows: enable-venv nombre_del_ambiente, En macOS y Linux: bin/enable-venv nombre_del_ambiente

Pregunta 5

1 / 1 pts

Considere el schema Pydantic siguiente:

from pydantic import BaseModel

```
class Item(BaseModel):
    name: str
    price: float
    quantity: int = 1
```

¿Cuál es el valor predeetterminado del campo `quantity`?

¡Correcto!

- (●) 1
- ( ) None
- ( ) El campo es requerido y no puede tener un valor predefinido
- ( ) 0

Pregunta 6

4 / 4 pts

¿Cómo se implementa la función dunder __getitem__ para acceder a elementos de una lista personalizada, llámela MiLista? (Muestre el código)

Para probar use lo siguiente:

```
mi_lista = MiLista([10, 20, 30, 40, 50])
```

```
print(mi_lista[20])  # Debería imprimir 3
```

Su respuesta:

```
class MiLista(list):
    ## Implementamos la funcion
    def __getitem__(self, indice):
        if indice >= len(self):
            raise IndexError("El indice no se encuentra dentro del rango.")
        return super().__getitem__(indice)


mi_lista = MiLista([10, 20, 30, 40, 50])
print(mi_lista[2])

#En este caso imprime el numero 30
```

Pregunta 7

4 / 5 pts

Crea una función generadora llamada potencias(numero, exponente), que devuelva las potencias sucesivas de un número dado, comenzando por 1 y elevando hasta un exponente especificado.

Para probar use lo siguiente:

```
potencias_2 = potencias(2, 5) # Generador de potencias de 2 hasta la 5ª for potencia in potencias_2: print(potencia) # Salida: 2 4 8 16 32
```

Su respuesta:

```
def potencias(numero, exponente):
    potencia = 1
    for i in range(exponente+1):
        yield potencia
        potencia *= numero

potencias_resultado = potencias(2, 5)
for potencia in potencias_resultado:
    print(potencia)

"""
En este caso la salida sería:
1
2
4
8
16
32
"""
```

1 no es potencia de 2

Pregunta 8

6 / 6 pts

Cree un aplicación simple con FastAPI que tenga cuatro endpoint tres GET y un
POST. En este caso va a usar un schema Pydantic llamado Usuario, el cual va
tener un id(int), un nombre completo(str), usuario(str), email (Pydantic.Email) y password (str). Los datos se guardaran en una lista que usará como base de datos.
El POST, incluirá un Usuario en la lista
El primer GET estará en la base de la aplicación e indicará un mensaje: "Bienvenidos a la aplicación de ejemplo"
El segundo GET mostrará todos los usuarios, en el path /users/
El segundo GET mostrará un usuario por su ID, en el path /users/{id}

Su respuesta:

Cuenta
Tablero
Cursos
Grupos
Calendario
Bandeja de entrada
Historial
Ayuda

Semestre Mar/Jul 2023-202...
Página de Inicio
Anuncios
Foros de discusión
Módulos
Evaluaciones

Quiz 1

Fecha de entrega 10 de abr en 13:00 **Puntos** 20 **Preguntas** 3
Disponible 10 de abr en 11:45 - 10 de abr en 13:00 1 hora y 15 minutos
Límite de tiempo 30 minutos

Detalles de la entrega:

Hora:	29 minutos
Puntaje actual:	19 de 20
se mantuvo el puntaje:	19 de 20

Instrucciones

Lea con detenimiento cada una de las preguntas que se le presentan y responda de acuerdo a lo que se le solicita.

Este examen fue bloqueado en 10 de abr en 13:00.

Historial de intentos

	Intento	Hora	Puntaje
MÁS RECIENTE	Intento 1	29 minutos	19 de 20

Puntaje para este examen: **19** de 20
Entregado el 10 de abr en 12:15
Este intento tuvo una duración de 29 minutos.

Pregunta 1

5 / 6 pts

Crea una clase base llamada Animal con atributos name y sound. Luego, crea subclases para tres (3) animales específicos (por ejemplo, Perro, Gato, Loro). Cada subclase debe sobrescribir el atributo sound y proporcionar un sonido específico para ese animal. Copie el código con su respuesta en el lugar indiccado

Su respuesta:

```
from abc import ABC, abstractmethod
#clase abstracta
class Animal(ABC):
    name:str
    sound:str

    @abstractmethod
    def make_sound(self):
        pass
#clases concreta
class Loro(Animal):
    def __init__(self):
        super().__init__('Loro')
        self.sound = 'romeo'
        self.name = 'loro'

    def make_sound(self):
        return self.sound
 #clases concreta
class Perro(Animal):
    def __init__(self):
        super().__init__('Perro')
        self.sound = 'guau'
        self.name = 'Perro'

    def make_sound(self):
        return self.sound

#clases concreta
class Gato(Animal):
    def __init__(self):
        super().__init__('Gato')
        self.sound = 'Miau'
        self.name = 'Gato'

    def make_sound(self):
        return self.sound
```

No había que hacer una clase abstracta

Pregunta 2

7 / 7 pts

En la página oficial de Python, en el carrusel de la página principal, existe un código para generar la serie de Fibonacci de forma iterativa con un ciclo while. Modifique la funcionalidad para que sea realizada con una función generadora, luego con un ciclo iterativo usando la función realizada entregue los primeros 15 números de la serie. Copia tu código con la respuesta en el lugar indicado.

Su respuesta:
#pagina de python:

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
```

#funcion generadora:

```
def fibonacci():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

fibonacciR = (x for x in fibonacci())
for i, number in enumerate(fibonacciR, 1):
    print(number)
    if i == 15:
        break
```

Pregunta 3

7 / 7 pts

La palabra **with** se usa en Python para gestionar un contexto, es así como se puede abrir un archivo usando with y no hay necesidad de cerrarlo, ya que el gestor se encarga de ello. Para crear un gestor de contexto se implemetan los métodos __enter__() y __exit__() para definir la entrada y salida respectivamente del contexto. Cree un clase llamada Timer, que aplicada a una iteración cualquiera mida el tiempo de ejecución de la iteración.

Ayuda: debe importar el módulo time y usar el método time() para obtener el tiempo actual de la máquina. Además en la implemnetación del metodo __exit__() debe tener un mensaje que indique el tiempo transcurrido en segundos. Para probar la funcionalidad de su clase hágalo con el siguiente código:

```
with Timer():
    # Bloque de código para medir el tiempo de ejecución
    for _ in range(1000000):
        pass
```

Copia tu código con la respuesta en el lugar indicado.

Su respuesta:

```
import time

class Timer:
    def __enter__(self):
        self.start_time = time.time()
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        end_time = time.time()
        elapsed_time = end_time - self.start_time
        print(f"Tiempo transcurrido: {elapsed_time:.6f} segundos")

# Prueba de la clase Timer
with Timer():
    for _ in range(1000000):
        pass
```

Puntaje del examen: **19** de 20

¿Cuáles son los métodos dunder que definen el funcionamiento de los siguientes operadores?: **>**, **and**, **in** y **%**, De una breve explicación de lo que hace cada uno.

Su respuesta:

Los métodos son:

__gt__(self, object): este método define al operador mayor igual (>), de manera que puede definirse condiciones especificas de comparación >.

__contains__(self, value): este método define al operador in, la cual es un operador de membresía que permite modificar el operador in para adecuarse a las condiciones específicas del caso.

__mod__(self,object): este método define el operador módulo (%), de manera que puede definirse condiciones especificas al utilizar dicha operación.

Cada uno de ellos permite definir las operaciones a las que corresponden para que se adapten al caso particular (enteros, cadena de caracteres, listas, etc)

¿y el operador and?

Pregunta 4

3 / 4 pts

Cree un esquema de Pydantic para representar un **usuario** de una aplicación. El esquema debe incluir los siguientes campos:

- Nombre: Cadena de texto, obligatorio, de longitud mínima de 2 caracteres
- Apellido: Cadena de texto, obligatorio, de longitud mínima de 3 caracteres
- Correo electrónico: Cadena de texto. obligatorio, máximo 80 caracteres
- Contraseña: Cadena de texto, obligatorio, minimo 6 caracteres

Su respuesta:

```
def Fibonacci(n):
    a=0
    b=a
    while a <= n:
        yield a
        print(a, end=' ')
        a, b = b, a+b


for value in Fibonacci(10):
    print(value)
```

Cae en un ciclo infinito, por no tener definida b = a, al comienzo, b debia ser = 1

n responder

Pregunta 6

0 / 6 pts

Cree una función decoradora que permita indicar cuál es el tiempo de ejecución que tarda en procesarse la función a la cual decora, Ayudas:

a. La función decoradora coloque el nombre **calcula_tiempo**

b. para usarla va a necesitar el módulo **time** de Python (importelo usando **import time**), el cual tiene una función llamada **time()** que devuelve el momento actual.

c. Para probar su decorador use el siguiente código:

```
import math
import time

@calcula_tiempo
def factorial(num):
    time.sleep(2)
    print(math.factorial(num))
```

d. pruebe llamado a la función factorial con un valor de **num =**

10, es decir **factorial(10)**

f. Al ejecutar el código del punto **d**, debe aparecer, en conjunto con el valor de factorial de 10, un mensaje que indique lo siguiente: **"El tiempo total en ejecutar fue de ___ segundos"**, donde el espacio en blanco se sustituirá por el valor calculado.

Su respuesta:

Puntaje del examen: **8** de 20

```
    OPERARIO = 3
    AYUDANTE = 4

class User(BaseModel):
    name: str = Field(min_length=2)
    lastname: str = Field(min_length=3)
    email: str = Field(max_length=80)
    password: str = Field(min_length=6)
    birthday: date
    rol: Optional[ROL]
```

El enum debia devolver un string no un entero, los campos name, lastname, email y password, debían ser obligatorios

Pregunta 5

1 / 6 pts

Cree una función generadora llamada **baraja()** que simule la entrega de cartas de un mazo, use las siguientes listas para definir el mazo de cartas:

valores = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']
palos = ['corazones', 'diamantes', 'tréboles', 'picas']

y para probar, ejecute el siguiente código:

```
i: int = 0
for carta in baraja():
    print(carta)
    i += 1
    if i > 5:
        break
```

La función debe entregar una tupla (valor, palo), por ejemplo ("3", "diamantes")

Su respuesta:

```
import random
def baraja():
```

```
    valores = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']
    palos = ['corazones', 'diamantes', 'tréboles', 'picas']
    yield (valores[random.randrange(0, len(valores))],
palos[random.randrange(0, len(valores))])

i: int = 0
for carta in baraja():
    print(carta)
    i += 1
    if i > 5:
        break
```

al ejecutarlo dió un list out of range. Revisando el código debía seleccionar el len(valores)-1

In responder

Pregunta 6

0 / 6 pts

Cree una función decoradora que permita indicar cuál es el tiempo de ejecución que tarda en procesarse la función a la cual decora, Ayudas:
a. La función decoradora coloque el nombre **calcula_tiempo**
b. para usarla va a necesitar el módulo **time** de Python (importelo usando **import time**), el cual tiene una función llamada **time()** que devuelve el momento actual.
c. Para probar su decorador use el siguiente código:

```
import math
import time

@calcula_tiempo
def factorial(num):
    time.sleep(2)
    print(math.factorial(num))
```

d. pruebe llamado a la función factorial con un valor de **num = 10,** es decir **factorial(10)**
f. Al ejecutar el código del punto **d**, debe aparecer, en conjunto con el valor de factorial de 10, un mensaje que indique lo siguiente: **"El tiempo**