Fundamentos de programación en PYTHON

Carrera de Administración de Redes y Comunicaciones

InGenio Learning

Docente: Mg. Gustavo Coronel

Alianzas Estratégicas

Microsoft Azure

aws educate

Google Meet

Cisco Networking Academy

Google Classroom

Kahoot!

nova cloud

ORACLE ACADEMY

FORTINET NSE Training Institute
Authorized Security Academy

RED HAT ACADEMY

CERTIPORT
A PEARSON VUE BUSINESS
AUTHORIZED TESTING CENTER

Pearson VUE
Authorized Test Center

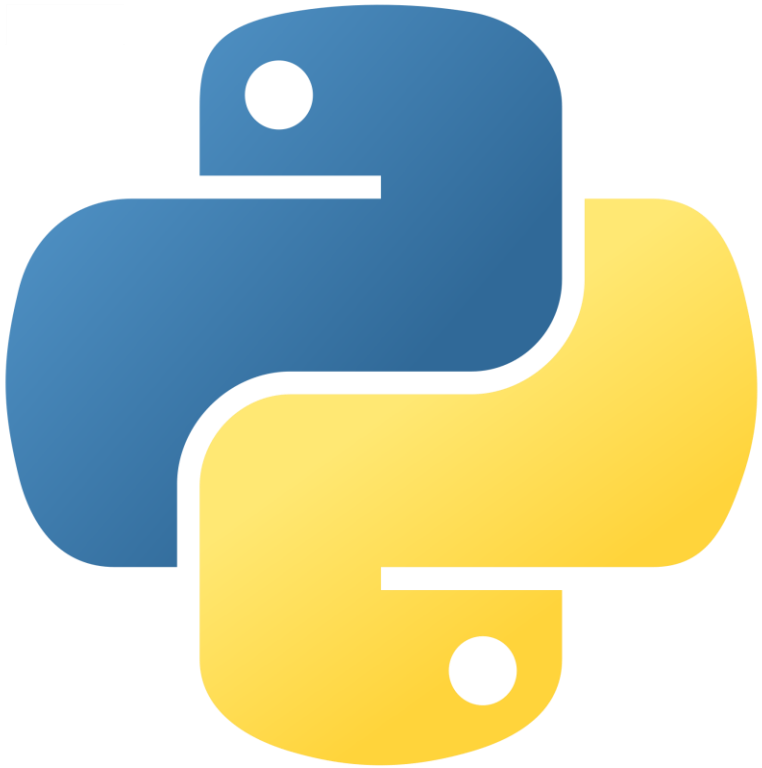
win E

Google Workspace

Office 365

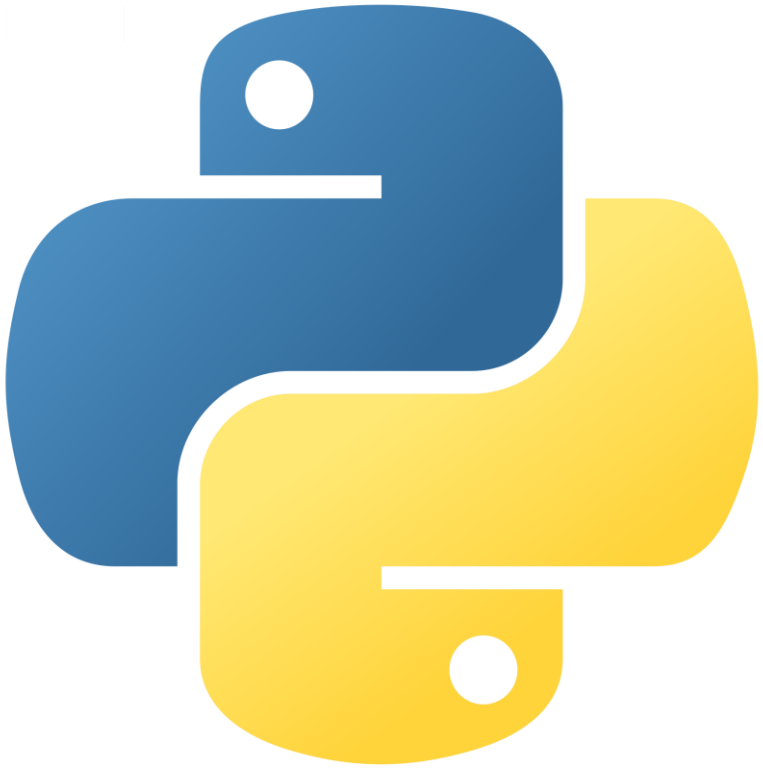
Q10
Académico

Sobre el curso



El curso está orientado a la enseñanza de fundamentos de Programación en el lenguaje Python v 3.0., bajo la modalidad definida por la Academia de Programming de Cisco. Esta modalidad consiste en el desarrollo de la currícula PCAP dividida en 8 módulos.

Durante el curso se desarrollarán actividades de revisión del contenido académico y laboratorios prácticos utilizando el emulador de Python que nos provee el mismo Cisco (SandBox) o de algún otro IDE para escritorio, así como el desarrollo de exámenes y actividades orientadas al entendimiento de estos conceptos de acuerdo con los temas que considera Cisco en este curso.



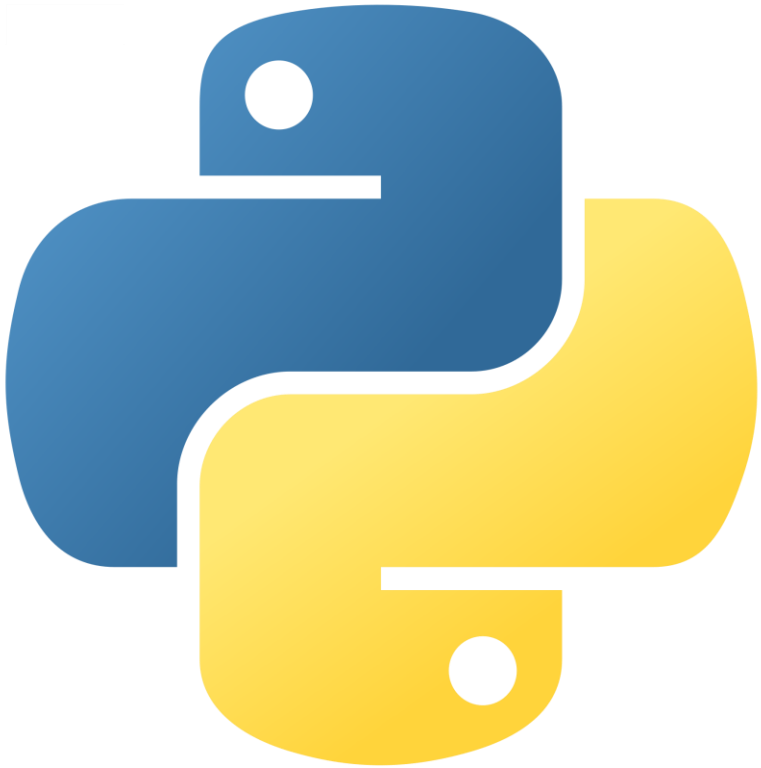
SEMANA 6

FUNCIONES

LOGRO DE LA SESION

Al finalizar la sesión, aprenderás a gestionar datos utilizando tuplas y diccionarios.

Contenido



- Introducción
- Definición
- Sintaxis
- Programando funciones
- Conclusiones
- Evaluación Continua 3

Introducción

Las tuplas y los diccionarios son dos tipos de estructuras de datos diferentes que se utilizan para almacenar y organizar información de manera eficiente.

TUPLA

```
( value1 , value2 , value3 )
```

DICCIONARIO

```
{ "nombre":"Gustavo",
  "curso":"SQL Server",
  "matriculados":24 }
```

Tuplas

Las tuplas son muy similares a las listas, pero con dos diferencias. Son inmutables, lo que significa que no pueden ser modificadas una vez declaradas, y en vez de inicializarse con corchetes se hace con ().

```
tupla = (10, 20, 30)
print(tupla)
print(tupla[1])
```

Tuplas

También pueden declararse sin (), separando los elementos por coma.

```
tupla = 10, 20, 30
print(tupla)
print(tupla[1])
```

Tuplas

Es posible crear una tupla de un solo elemento, para lo cual debes usar una coma antes del paréntesis de cierre.

```
notas = (15,)
print(notas)
print(notas[0])
```

Tuplas

Las tuplas también se pueden iterar.

```
ciudades = ("Chiclayo","Lima","Arequipa")
for ciudad in ciudades:
    print(ciudad)
```

Tuplas

Se puede también asignar el valor de una tupla con **"n"** elementos a **"n"** variables.

```
ciudades = ("Chiclayo","Lima","Arequipa")
ciudad1, ciudad2, ciudad3 = ciudades
print(ciudad1)
print(ciudad2)
print(ciudad3)
```

Métodos de Tuplas

FUNCIÓN	EJEMPLO	RESPUESTA
Count(elemento) Cuenta el número de veces que un elemento aparece em la tupla.	`tupla = [10,20,40,20,30,20]` `print(tupla.count(20))`	3
index(<elemento>[,index]) Busca un elemento y devuelve el índice en el que se ha encontrado.	`tupla = [10,20,40,20,30,20]` `print(tupla.index(40))`	2

Diccionarios

Un diccionario es una colección de elementos, donde cada uno tiene una llave y un valor.

```
prod = {
    "Nombre": "Televisor",
    "precio": 3674.89,
    "stock": 54
}
print(prod)
```

Diccionarios

Un diccionario es una colección de elementos, donde cada uno tiene una llave y un valor.

```
prod = dict([
    ("Nombre", "Televisor"),
    ("precio", 3674.89),
    ("stock", 54)
])
print(prod)
```

Diccionarios

También es posible usar el constructor dict() para crear un diccionario.

```
prod = dict(
    nombre = "Televisor",
    precio = 3674.89,
    stock = 54
)
print(prod)
```

Diccionarios

Propiedades de los diccionario en Python son las siguientes:

- Son **dinámicos**, pueden crecer o decrecer, se pueden añadir o eliminar elementos.
- Son **indexados**, los elementos del diccionario son accesibles a través de la clave.
- Y son **anidados**, un diccionario puede contener a otro diccionario .

```
prod = dict(
    nombre = "Televisor",
    precio = 3674.89,
    stock = 54
)
print(prod)
```

Diccionarios

Se puede acceder a los elementos con [] o también con la función get().

```
prod = dict(
    nombre = "Televisor",
    precio = 3674.89,
    stock = 54
)
print(f"Nombre: {prod["nombre"]}")
print(f"Precio: {prod.get("precio")}")
```

Diccionarios

Para modificar un elemento basta con usar [] con la clave y asignar el nuevo valor.

```
prod = dict(
    nombre = "Televisor",
    precio = 3674.89,
    stock = 54
)
prod["nombre"] = "Laptop"
print("Nombre:",prod["nombre"])
```

Diccionarios

Si la clave no existe, se añade automáticamente.

```
prod = dict(
  nombre = "Televisor",
  precio = 3674.89,
  stock = 54
)
prod["tamanio"] = "40 pulgadas"
print(prod)
```

Diccionarios

Iterando las claves de un diccionario.

```
prod = dict(
    nombre = "Televisor",
    precio = 3674.89,
    stock = 54
)
for clave in prod:
    print(clave)
```

Diccionarios

Mostrando los valores de un diccionario.

```
prod = dict(
    nombre = "Televisor",
    precio = 3674.89,
    stock = 54
)
for clave in prod:
    print(prod[clave])
```

Diccionarios

Iterando la clave y el valor de un diccionario.

```
prod = dict(
    nombre = "Televisor",
    precio = 3674.89,
    stock = 54
)
for clave,valor in prod.items():
    print(clave,":",valor)
```

Métodos de Diccionarios

FUNCIÓN	EJEMPLO	RESPUESTA

clear()
Elimina todo el contenido del diccionario.
	`prod = dict(`
	`  nombre = "Televisor",`
	`  precio = 3674.89,`
	`  stock = 54`
	`)`
	`prod.clear()`
	`print(prod)`
		`{}`

Métodos de Diccionarios

FUNCIÓN	EJEMPLO	RESPUESTA

get(<key>[,<default>])
Permite consultar el valor para un clave especifica.
El segundo parámetro es opcional, y en el caso de proporcionarlo es el valor a devolver si no se encuentra la key.
	`prod = dict(`
	`  nombre = "Televisor",`
	`  precio = 3674.89,`
	`  stock = 54`
	`)`
	`print(prod.get("nombre"))`
		`Televisor`

Métodos de Diccionarios

FUNCIÓN	EJEMPLO

items()
Devuelve una lista con las claves y valores del diccionario.
Si se convierte en lista se puede indexar como una lista, siendo los primeros elementos las claves y los segundos los valores.
	`d = dict(`
	`  a = 10,`
	`  b = 20,`
	`  c = 30`
	`)`
	`it = d.items()`
	`print(it)`
	`print(list(it))`

Métodos de Diccionarios

FUNCIÓN	EJEMPLO

keys()
Devuelve una lista con todas las claves del diccionario.
	`prod = dict(`
	`  nombre = "Televisor",`
	`  precio = 3674.89,`
	`  stock = 54`
	`)`
	`k = prod.keys()`
	`print(k)`
	`print(list(k))`

Métodos de Diccionarios

FUNCIÓN	EJEMPLO

values()
Devuelve una lista con todos los valores del diccionario.
	`prod = dict(`
	`  nombre = "Televisor",`
	`  precio = 3674.89,`
	`  stock = 54`
	`)`
	`k = prod.values()`
	`print(k)`
	`print(list(k))`

Conclusiones

- Los diccionarios son útiles cuando necesitas almacenar información que quieres poder recuperar rápidamente utilizando una clave.
- Mientras que las tuplas son más adecuadas cuando necesitas una secuencia ordenada e inmutable de elementos.

Evaluación continua

Desarrollar los problemas propuestos para esta semana


GRACIAS
TOTALES