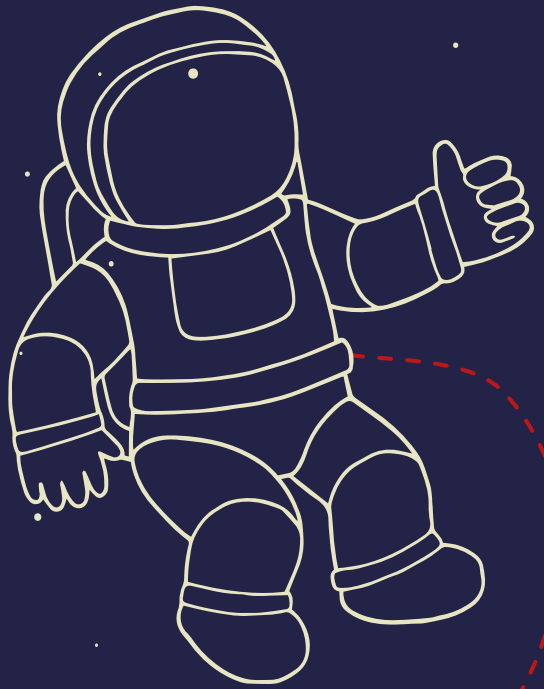
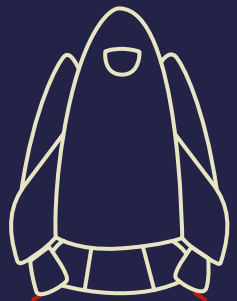
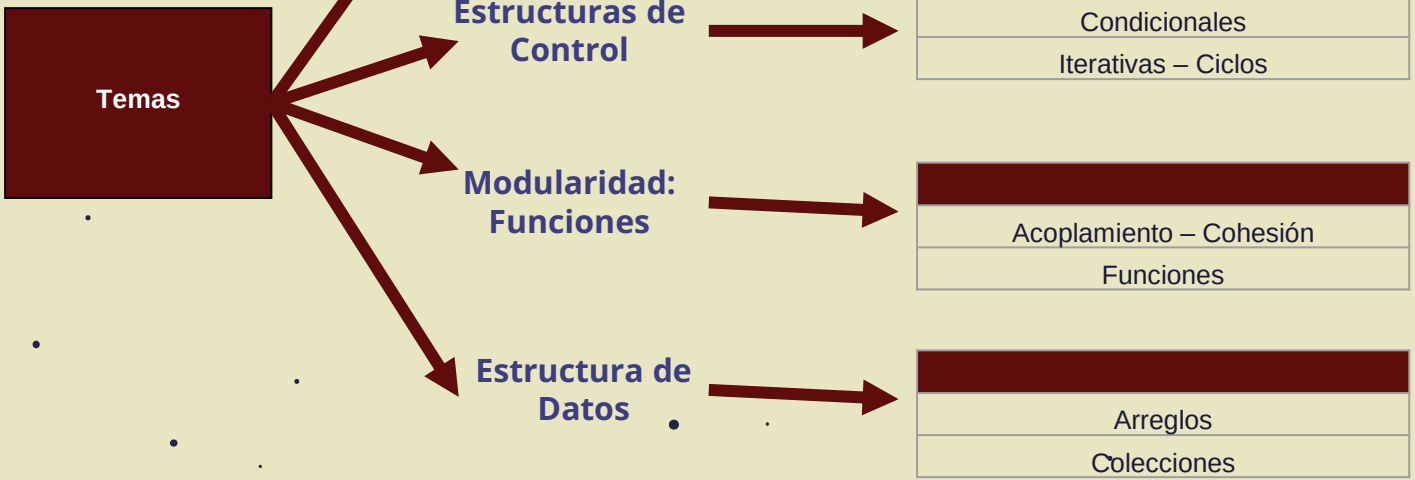


Programa académico CAMPUS

Ciclo 1: Fundamentos de Programación



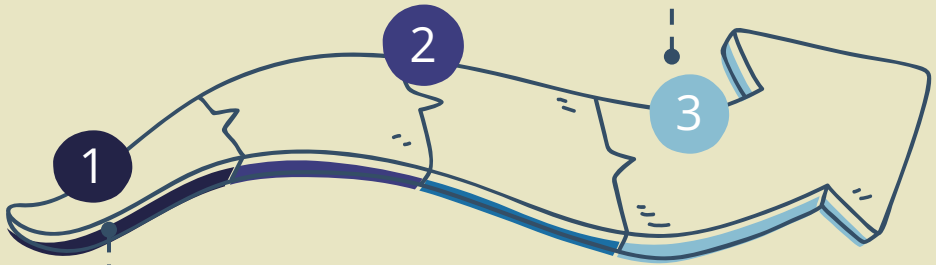
Presentación Ciclo 1 – Fundamentos de Programación



Estructura de Datos

Diccionarios

Taller



Listas

Estructura de Datos

Conceptualización

En la vida cotidiana nos vemos enfrentados a crear listas, por ejemplo la lista de útiles para el colegio o la universidad, la lista de personas que se invitará a una fiesta. En la preparación de alimentos, se debe realizar una lista con los ingredientes: carne molida, tomate, pan, cebolla, aceite, queso tajado, lechuga y tocineta.

Las **estructuras de datos** son agrupaciones de variables simples que conforman un conjunto de datos más complejo con el cual puedes dar soluciones eficientes a situaciones más cercanas a la vida práctica, como lo son por ejemplo: el manejo de calificaciones de estudiantes de un curso o la gestión de nómina de los empleados de una empresa.

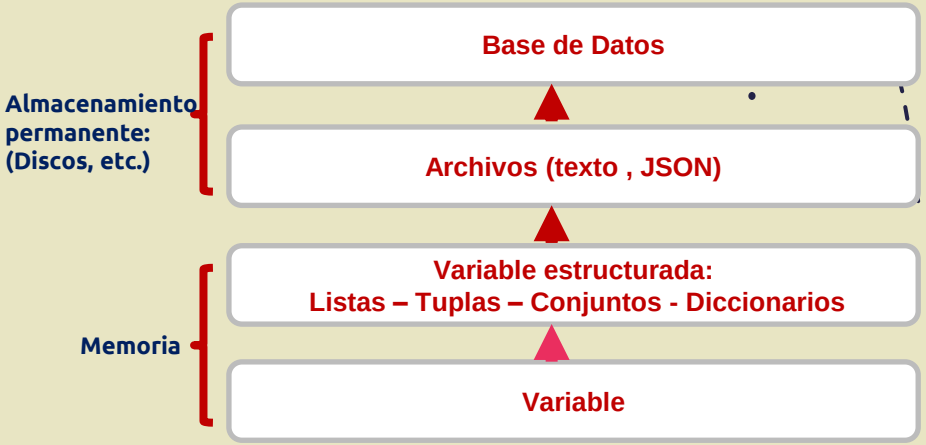
Estructura de Datos

Conceptualización

Variable: Es un espacio de memoria que **contiene un dato simple** de tipo cadena, numérico, booleano, etc.

Al contenido de este espacio de memoria, se accede a través de lo que llamamos un **identificador o Nombre de Variable.**

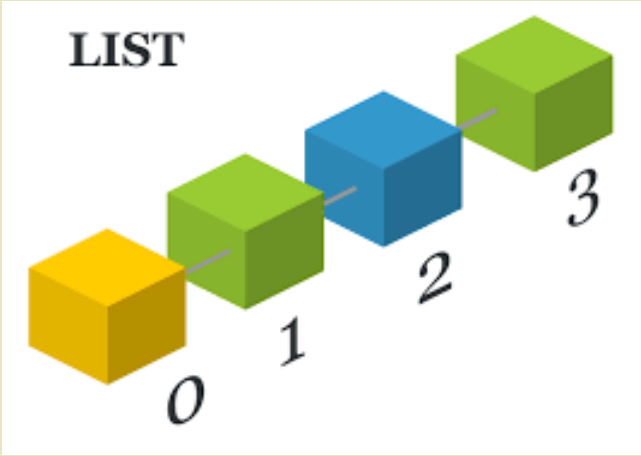
Variable Estructurada: es un agrupamiento, empaquetamiento o colección de varios espacios de memoria, a los cuales se accede a través de un único identificador**.**



Estructura de Datos

Listas

Las listas son estructuras que permiten ser modificadas a lo largo de la ejecución de un programa usando algunos métodos y operadores. Este comportamiento le da la caracterización a las listas de ser **estructuras mutables**.



Estructura de Datos

Listas - Características

Referenciar elementos

nombre_persona[2] = **“Sergio”**

nombre_persona[0:2] = **“Catalina”,”Silvia”**

Estructura de Datos

Listas - Creación

```
>>> lista_numeros=[10,15,20,30,40]
>>> lista_numeros[2]
20
>>> lista_nombres=["Sergio","Catalina","Silvia","Iván","Elsa"]
>>> lista_nombres[4]
'Elsa'
```

```
>>> lista_nombres[0:2]
['Sergio', 'Catalina']
```

```
>>> lista_pares=list(range(2,20,2))
>>> lista_pares
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Crea la lista con valores desde 2 hasta 20, con incrementos de 2. No se toma el valor final del rango

```
>>> lista_elementos=[1,"Juan",[2,3],10.4,"Pedro"]
>>> lista_elementos[2]
[2, 3]
>>> lista_elementos[3:5]
[10.4, 'Pedro']
```

Estructura de Datos

Listas - Métodos

Podemos crear una lista y luego modificar sus elementos mientras se ejecuta el código.

Para esto, las listas tienen un conjunto de métodos y funciones que realizan acciones y operaciones sobre una lista en particular. Algunos de estos métodos son:

append, extend, insert, pop, remove y otros como len para la longitud, es decir número de elementos

El método append permite añadir un ítem al final de una lista

El método extend se utiliza para agregar elementos interables como un string u otra lista separando sus elementos.

El método insert permite añadir un ítem en una posición o índice específico

El método pop quita un elemento de la lista dado su índice.

El método remove para remover un ítem de una lista basado en el valor

Estructura de Datos

Listas - Métodos

```
>>> lista=[10,20,"Juan",30,"Sergio"]
>>> lista
[10, 20, 'Juan', 30, 'Sergio']
>>> lista.append(40)
>>> lista
[10, 20, 'Juan', 30, 'Sergio', 40]
>>> lista.append("Paula")
>>> lista
[10, 20, 'Juan', 30, 'Sergio', 40, 'Paula']
>>> lista.extend([60,80])
>>> lista
[10, 20, 'Juan', 30, 'Sergio', 40, 'Paula', 60, 80]
>>> lista.insert(1,"Luis")
>>> lista
[10, 'Luis', 20, 'Juan', 30, 'Sergio', 40, 'Paula', 60, 80]
>>> lista.pop(4)
30
>>> lista
[10, 'Luis', 20, 'Juan', 'Sergio', 40, 'Paula', 60, 80]
>>> lista.remove("Sergio")
>>> lista
[10, 'Luis', 20, 'Juan', 40, 'Paula', 60, 80]
```

Estructura de Datos

Funciones predefinidas en Python para listas

En los lenguajes de programación, se puede definir cualquier función que se desee; pero, para facilitarnos un poco la vida existen funciones predefinidas, o sea que alguien más ya las creó y fueron incorporadas en el lenguaje de programación, en este caso **Python**.

En **Python**, se pueden encontrar dos tipos de funciones predeterminadas, las cuales son:

- **Funciones predefinidas**.
- **Los módulos predefinidos**. son archivos que contienen **métodos predefinidos** (funciones), que no pueden existir por sí solos, ya que se encuentran asociados a determinado **objeto** o tipo de dato (listas, cadenas, caracteres, etc), de tal manera que, operan sobre ellos.

Estructura de Datos

Funciones predefinidas en Python para listas

```
>>> max(40,30,5,90,20)
90
```

```
>>> min(40,30,5,90,20)
5
```

Funciones

Librerías: Agrupan , funciones, métodos, etc.

```
>>> numeros=[11,12,31,41,52,63,78]
>>> len(numeros)
7
```

```
>>> import math
>>> math.sqrt(16)
4.0
```

Método: Operación

```
>>> numeros=[6,3,1,4,5,2]
>>> numeros_ordenados=sorted(numeros)
>>> numeros_ordenados
[1, 2, 3, 4, 5, 6]
```

```
>>> numeros=[6,3,1,4,5,2]
>>> numeros_ordenados=sorted(numeros,reverse=True)
>>> numeros_ordenados
[6, 5, 4, 3, 2, 1]
```

Estructura de Datos

Funciones predefinidas en Python – Programa

```
# Programa para hallar la raíz cuadrada de un número
# Autor: Sergio Medina
# Fecha: 09/05/2022

import math
x=int(input("Valor: "))
print("Raiz cuadrada de ",x," es: ",math.sqrt(x))
```

Estructura de Datos

Funciones predefinidas en Python para listas

```
>>> numeros=[1,2,3,4,5,6,1,7,1]
>>> print(numeros.count(1))
3
```

```
>>> letras=["a","e","t","v","u","Z","c","a"]
>>> print(letras.count("a"))
2
```

Contar en listas

```
>>> items="1,2,3,4,5,6"
>>> items.split(",")
['1', '2', '3', '4', '5', '6']
>>> items.split(",")[4]
'5'
```

```
>>> items="la casa de Luisa es muy bonita"
>>> items.split()
['la', 'casa', 'de', 'Luisa', 'es', 'muy', 'bonita']
```

Crea Lista desde una cadena , su argumento es el separador de elementos de la lista creada

Cantidad de palabras en una frase

```
>>> items="la casa de Luisa es muy bonita"
>>> items.split()
['la', 'casa', 'de', 'Luisa', 'es', 'muy', 'bonita']
>>> len(items.split())
7
```

Estructura de Datos

Funciones predefinidas en Python – Programa

```
# Programa para calcular la cantidad de palabras de un frase
# Autor: Sergio Medina
# Fecha: 10/05/2022

frase=input("Frase: ")
can_palabras=len(frase.split())
print("Cantidad de palabras: ",can_palabras)
```

Estructura de Datos

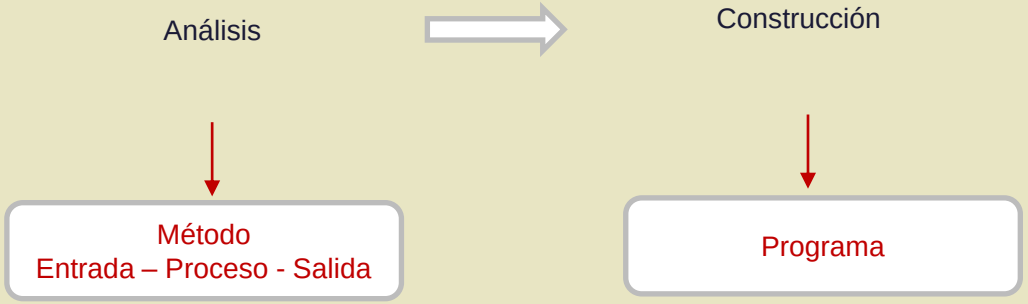
Listas – Ejercicio – Contar Vocales

- Se desea realizar un programa en el cual se ingresen N letras del abecedario, las cuales se deben almacenar en una lista. Una vez creada la lista, se desea conocer e imprimir la cantidad de “a”, la cantidad de “e”, la cantidad de “i”, la cantidad de “o” y la cantidad de “u” que se encuentran en la lista

Estructura de Datos

Listas - Ejercicio

Metodología -> Pensamiento lógico estructurado



Estructura de Datos

Listas - Ejercicio

1 2
3 4
5 6

Análisis –> Ejercicio Listas

Lista & Ciclos

- Llenar
- Procesar

Entrada LEER → Proceso → Salida IMPRIMIR

Estructura de Datos

Listas - Ejercicio

```
# Programa para contar vocales de una lista
# Autor: Sergio Medina
# Fecha: 08/06/2022

#iniciar lista como vacias
lista_letras=[]
N=int(input("Cantidad de letras: "))
cantidad_a=0
cantidad_e=0
cantidad_i=0
cantidad_o=0
cantidad_u=0
#LLenar lista
for i in range(N):
    letra=input("Letra: ")
    lista_letras.append(letra)
print("Lista letras: ",lista_letras)
#Procesar lista
for x in lista_letras:
    if x=="a" or x=="A":
        cantidad_a+=1
    elif x=="e" or x=="E":
        cantidad_e+=1
    elif x=="i" or x=="I":
        cantidad_i+=1
    elif x=="o" or x=="O":
        cantidad_o+=1
    elif x=="u" or x=="U":
        cantidad_u+=1
print("Cantidad de a: ",cantidad_a)
print("Cantidad de e: ",cantidad_e)
print("Cantidad de i: ",cantidad_i)
print("Cantidad de o: ",cantidad_o)
print("Cantidad de u: ",cantidad_u)
```

Estructuras de Datos

Diccionarios

Los diccionarios son estructuras de datos que permiten almacenar **valores** indexados, a través de **claves,** lo cual permite ordenar datos a través de la clave y realizar una búsqueda más eficiente

Para crear un diccionario, empaquetamos entre llaves { } cada par de elementos **Clave:Valor** separadas por comas, así:

```
empleado = {'nombre':"Sergio", 'apellido':"Medina", 'edad':57, 'salario':3500000}
```

Estructuras de Datos

Diccionarios - Crearlos


```
d1 = {
  "Nombre": "Sara",
   "Edad": 27,
   "Documento": 1003882
}
print(d1)
#{'Nombre': 'Sara', 'Edad': 27, 'Documento':
1003882}
```

Estructuras de Datos

Diccionarios - Crearlos

Otra forma equivalente de crear un diccionario en Python es usando `dict()` e introduciendo los pares `key:value` entre paréntesis.

Estructuras de Datos

Diccionarios - Crearlos


```
d2 = dict([
    ('Nombre', 'Sara'),
    ('Edad', 27),
    ('Documento', 1003882),
])
print(d2)
#{'Nombre': 'Sara', 'Edad': '27',
'Documento': '1003882'}
```

Estructuras de Datos

Diccionarios - Crearlos

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


```
d3 = dict(Nombre='Sara',
          Edad=27,
          Documento=1003882)
print(d3)
#{'Nombre': 'Sara', 'Edad': 27, 'Documento': 1003882}
```

Estructuras de Datos

Diccionarios -Acceso

Una vez que almacenamos los datos en el diccionario, vamos a acceder a ellos

Con el **método get()** de un diccionario, podemos obtener el valor de una clave, pero si no existe la clave devolver un mensaje en string como respuesta.

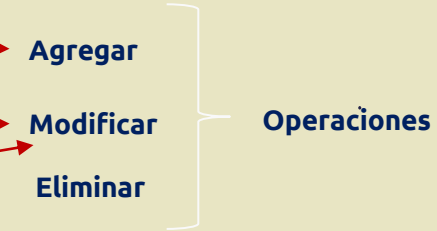
```
>>> empleado={'nombre':"Sergio Medina",'cargo':"Programador",'salario':4000000}
>>> empleado
{'nombre': 'Sergio Medina', 'cargo': 'Programador', 'salario': 4000000}
>>> empleado['nombre']
'Sergio Medina'
>>> empleado.get('nombre')
'Sergio Medina'
>>> empleado['email']
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    empleado['email']
KeyError: 'email'
>>> empleado.get('email',"NO ENCONTRADO")
'NO ENCONTRADO'
```

Cuando la clave NO es encontrada en el diccionario

Estructuras de Datos

Diccionarios - Operaciones

```
>>> articulos={1:"Lapiz",2:"Borrador",3:"Cuadernos"}
>>> articulos
{1: 'Lapiz', 2: 'Borrador', 3: 'Cuadernos'}
>>> articulos[4]="Calcualdora"
>>> articulos
{1: 'Lapiz', 2: 'Borrador', 3: 'Cuadernos', 4: 'Calcualdora'}
>>> articulos[4]="Calculadora"
>>> articulos
{1: 'Lapiz', 2: 'Borrador', 3: 'Cuadernos', 4: 'Calculadora'}
>>> articulos[5]="Refresco"
>>> articulos
{1: 'Lapiz', 2: 'Borrador', 3: 'Cuadernos', 4: 'Calculadora', 5: 'Refresco'}
>>> del articulos[5]
>>> articulos
{1: 'Lapiz', 2: 'Borrador', 3: 'Cuadernos', 4: 'Calculadora'}
```



Estructuras de Datos

Diccionarios - Iterar

Los diccionarios se pueden iterar de manera muy similar a las listas u otras estructuras de datos. Para imprimir los key.

```
# Imprime los key del diccionario
#d1:{'Nombre':'Laura', 'Edad’:27, 'Documento':1003882,
'Direccion': 'Calle 123'}
for x in d1:
    print(x)
#Nombre
#Edad
#Documento
#Direccion
```

Estructuras de Datos

Diccionarios - Iterar

Se puede imprimir también solo el value.

```
# Imprime los value del diccionario
for x in d1:
        print(d1[x])
#Laura
#27
#1003882
#Calle 123
```

Estructuras de Datos

Diccionarios - Iterar

O si queremos imprimir el key y el value a la vez.

```
# Imprime los key y value del diccionario
for x, y in d1.items():
        print(x, y)
#Nombre Laura
#Edad 27
#Documento 1003882
#Direccion Calle 123
```

Estructuras de Datos

Diccionarios - Algunos métodos más usados

clear()	>>> **versiones** = dict(**python**=2.7, **zope**=2.13, **plone**=5.1)
	>>> print **(versiones)**
	{'python': 2.7, 'zope': 2.13, 'plone': 5.1}
	>>> **versiones.clear()**
	>>> print(**versiones**)
	{ }

copy()	>>> versiones = dict(python=2.7, zope=2.13, plone=5.1)
	>>> otro_versiones = versiones.copy()
	>>> versiones == otro_versiones
	True

get()	>>> versiones = dict(python=2.7, zope=2.13, plone=5.1)
	>>> versiones.get('plone')
	5.1

pop()	Recibe como parámetro una clave, elimina esta y devuelve su valor. Si no lo encuentra, devuelve error. dic = {'a' : 1, 'b' : 2, 'c' : 3 , 'd' : 4} valor = **dic.pop**('b') valor → 2 dic → {'a' : 1, 'c' : 3 , 'd' : 4}

keys()	>>> versiones = dict(python=2.7, zope=2.13, plone=5.1)
	>>> versiones.keys()
	['zope', 'python', 'plone']

Estructuras de Datos

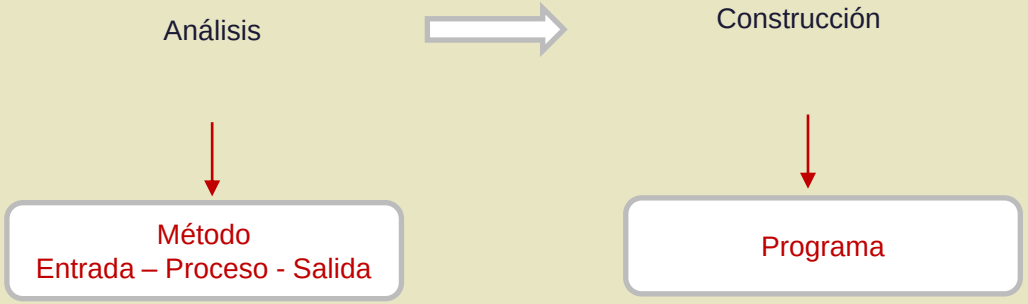
Diccionarios -Ejercicio

Contar el número de veces que una palabra aparece dentro de un texto.

Estructuras de Datos

Diccionarios -Ejercicio

Metodología -> Pensamiento lógico estructurado



Estructuras de Datos

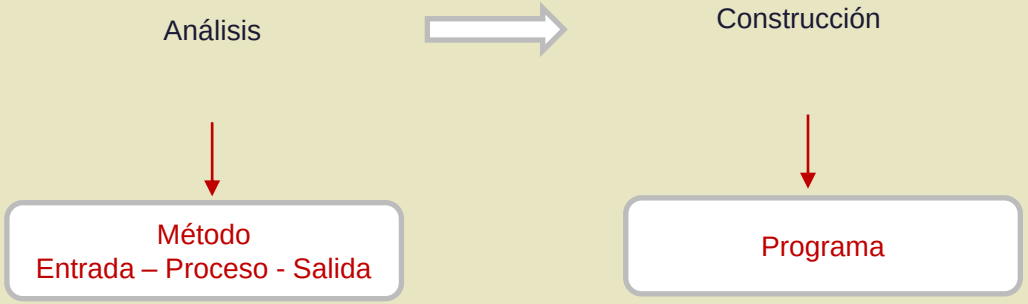
Diccionarios -Ejercicio

Solicitar el id, el nombre, el apellido y la fecha de nacimiento al usuario hasta que este ingrese como id un número negativo. Esta información debes agregarla a un diccionario. Por último crear, como salida, un informe donde se aprecie la información digitada de manera más legible.

Estructuras de Datos

Diccionarios -Ejercicio

Metodología -> Pensamiento lógico estructurado



Estructuras de Datos

Diccionarios -Ejercicio

La institución educativa “SamEduca” cuenta con N docentes, conociendo de cada uno de ellos su número de cédula, nombre, categoría y números de horas laboradas en el mes. Se pide realizar un programa que calcule el valor de los honorarios de cada docente y el valor total a pagar por concepto de honorarios. Para este proceso, nos suministran el diccionario donde se define el valor de la hora para cada categoría, así:

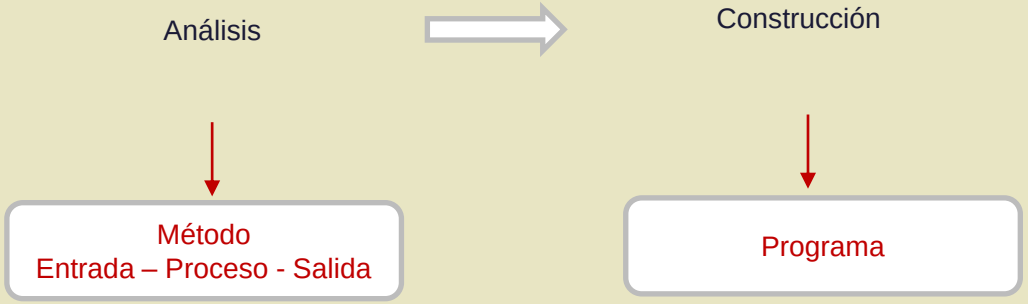
diccionario_categoria={1:25000,2:30000,3:40000,4:45000,5:60000}

Se debe imprimir el nombre del docente, el valor de sus honorarios y el valor total de honorarios, el de los N docentes.

Estructuras de Datos

Diccionarios -Ejercicio

Metodología -> Pensamiento lógico estructurado



Estructuras de Datos

Diccionarios -Ejercicio

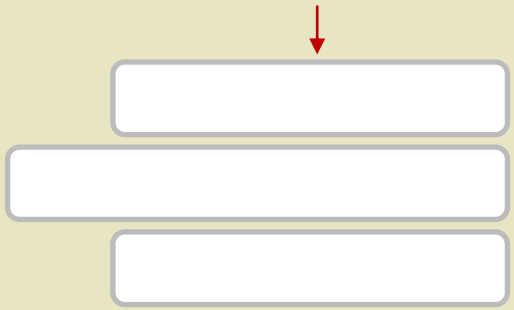
1 2 3 4 5 6 7

Análisis –> Ejercicio Listas

Entrada LEER

Proceso

Salida IMPRIMIR



Estructuras de Datos

Diccionarios –Ejercicio – Versión 1

```
# Programa para manejo de diccionarios
# Autor: Sergio Medina
# Fecha: 14/06/2022

diccionario_categoria={1:25000,2:30000,3:40000,4:45000,5:60000}
N=int(input("Cantidad de docentes: "))
total_honorarios=0
for i in range(N):
    cedula=int(input("Cédula docente: "))
    nombre=input("Nombre docentes: ")
    categoria=int(input("Categoría: "))
    horas=int(input("Horas laboradas en mes: "))
    honorarios=horas*diccionario_categoria.get(categoria)
    total_honorarios+=honorarios
    print("Nombre docente: ",nombre)
    print("Honorarios: ",honorarios)
print("Total Honorarios: ",total_honorarios)
```

Estructuras de Datos

Diccionarios –Ejercicio – Versión 2 (Validación)

```
# Programa para manejo de diccionarios
# Autor: Sergio Medina
# Fecha: 14/06/2022

diccionario_categoria={1:25000,2:30000,3:40000,4:45000,5:60000}
N=int(input("Cantidad de docentes: "))
total_honorarios=0
for i in range(N):
    cedula=int(input("Cédula docente: "))
    nombre=input("Nombre docentes: ")

    while True:
        try:
            categoria=int(input("Categoría: "))
            if diccionario_categoria.get(categoria,"ERROR")=="ERROR":
                print("Categoría NO Existe")
                continue
            break
        except ValueError:
            print("La categoría debe ser un dato entero")

    horas=int(input("Horas laboradas en mes: "))
    honorarios=horas*diccionario_categoria.get(categoria)
    total_honorarios+=honorarios
    print("Nombre docente: ",nombre)
    print("Honorarios: ","{:,.2f}".format(honorarios))
print("Total Honorarios: ","{:,.2f}".format(total_honorarios))
```

Talleres:

Realizar los ejercicios usando arreglos y colecciones

Estructura de Datos

Escribir un programa que guarde en un diccionario los precios de las frutas de la tabla, pregunte al usuario por una fruta, un número de kilos y muestre por pantalla el precio de ese número de kilos de fruta. Si la fruta no está en el diccionario debe mostrar un mensaje informando de ello.

Fruta	Precio
Plátano	1.35
Manzana	0.80
Pera	0.85
Naranja	0.70

Estructura de Datos

Situación Problema: Pares e Impares

Se desea realizar un programa en el cual se ingresen números enteros, los cuales se deben almacenar en una lista. Se debe ingresar números hasta que el número ingresado sea 99999. Una vez creada la lista, se desea conocer cuales y cuántos son pares e impares.

Estructura de Datos

Situación Problema: Contar palabras

Dada una lista con nombres completos de personas, realizar un programa que genere una segunda con la cantidad de palabras de cada uno de los nombres. La lista de nombres debe llenarse a través de nombres que se ingresan por teclado, hasta que el nombre ingresado sea “FIN”

Se debe imprimir la lista de nombres y la lista con la cantidad de palabras de cada nombre.

Estructura de Datos

Escribir un programa que gestione las facturas pendientes de cobro de una empresa. Las facturas se almacenarán en un diccionario donde la clave de cada factura será el número de factura y el valor el coste de la factura. El programa debe preguntar al usuario si quiere añadir una nueva factura, pagar una existente o terminar. Si desea añadir una nueva factura se preguntará por el número de factura y su coste y se añadirá al diccionario. Si se desea pagar una factura se preguntará por el número de factura y se eliminará del diccionario. Después de cada operación el programa debe mostrar por pantalla la cantidad cobrada hasta el momento y la cantidad pendiente de cobro.

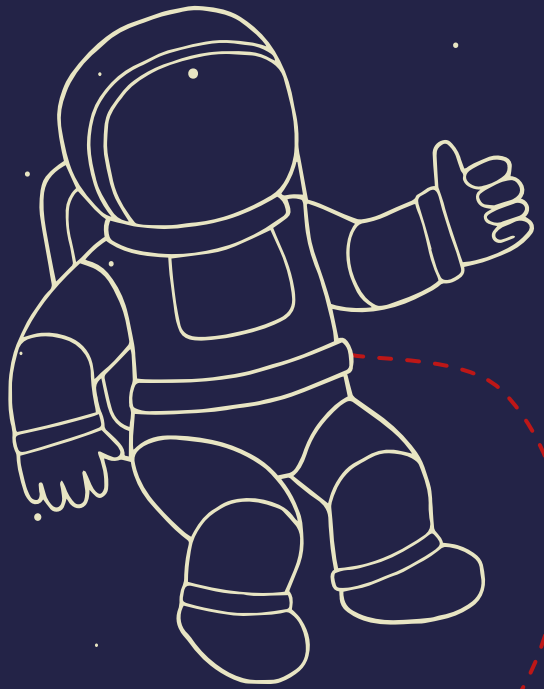
Estructura de Datos

Se realiza la compra de N artículos, en donde se ingresa el código del artículo y la cantidad y mediante el uso de diccionarios para los nombres y valores unitarios de los artículos, el programa debe obtener el nombre de cada artículo, cantidad comprada, valor unitario, valor total de acuerdo a la cantidad comprada y finalmente calcular el valor total de la compra.

Se suministra el diccionario de nombres de artículo y otro con los valores unitarios.

articulos={1:"Lapiz",2:"Cuadernos",3:"Borrador",4:"Calculadora",5:"Escuadra"}

valores={1:2500,2:3800,3:1200,4:35000,5:3700}



Programa académico CAMPUS

Ciclo 1: Fundamentos de Programación