FACULTAD DE INGENIERÍA
UNICEN • OLAVARRÍA

Fundamentos de la Programación

y

Métodos Numéricos

CLASE 2.

Estructuras de flujo de control básicas: repetitivas o ciclicas. Estructuras de datos. Introducción a la Modularización: estrategia, diagrama de estructura, parámetros, funciones en Python.

SEMANA PASADA

Juego: Piedra-Papel-Tijera

Juego: Piedra-Papel-Tijera

SEMANA PASADA

- Usuario Vs. Maquina
- La maquina selecciona una opción entre tres posibles opciones: Piedra, Papel o Tijera (opcion_maquina).
- El usuario selecciona una opción entre tres posibles opciones: Piedra, Papel o Tijera (opcion_usuario).
- Si la opción del usuario es igual a la de la opción maquina, entonces el resultado es empate.
- Para que el usuario gane:

Usuario	Máquina
Piedra	Tijera
Tijera	Papel
Papel	Piedra

Juego: Piedra-Papel-Tijera

SEMANA PASADA

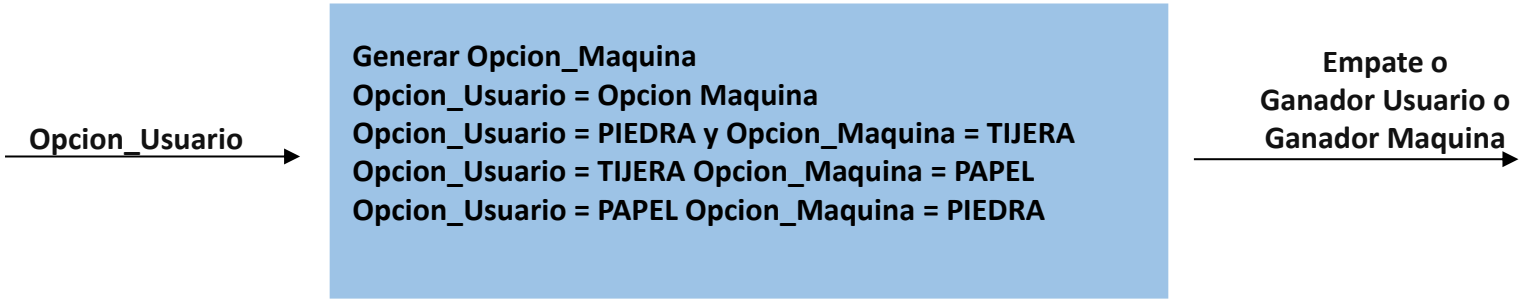
2. Análisis del Problema: se basa en identificar con claridad tres cosas:

- ¿Qué **entradas** se requieren y la forma que los mismos llegan al programa?
- ¿Cuál es la **salida** deseada y de que manera debe presentarse?
- ¿Qué **tratamiento (método)** ha de realizarse a los datos de entrada para producir la salida deseada?

La entrada requerida:
Opción del usuario (Piedra (1) /Papel (2)/Tijera (3))

La salida esperada:
Empate/Ganador Usuario/Ganador Maquina

El programa debe:
Generar la opción de la maquina y comparar opciones del usuario y la maquina para determinar quien gana



Juego: Piedra-Papel-Tijera

SEMANA PASADA

3. DISEÑO DEL ALGORITMO SOLUCION

Piedra_Papel_Tijera

#Por convención: 1:Piedra; 2:Papel; 3:Tijera

1. Generar aleatoriamente un valor entre 1 y 3 que sea la opción de la maquina y guardar en la variable **maquina**
2. Leer la opción del usuario y guardarlo en la variable **usuario**
3. Si usuario = maquina entonces mostrar en pantalla “Resultado: empate”
4. Sino Si (usuario = 1 AND maquina=3) OR (usuario = 2 AND maquina=1) OR (usuario = 3 AND maquina=2) entonces mostrar en pantalla “Resultado: gana el usuario”
5. Sino imprimir en pantalla “Resultado: gana la maquina”

Para que el usuario gane:

Usuario	Maquina
Piedra (1)	Tijera (3)
Tijera (3)	Papel (2)
Papel (2)	Piedra (3)

Juego: Piedra-Papel-Tijera

SEMANA PASADA

5. CODIFICACIÓN DEL ALGORITMO EN PYTHON

main.py +

```
1
2  # Online Python - IDE, Editor, Compiler, Interpreter
3
4  import random
5  #1:Piedra ;  2:papel; 3:tijera
6
7  maquina=random.randint(1, 3)
8  usuario = int(input('Ingrese su opcion; PIEDRA (1) / PAPEL (2) / TIJERAS (3): '))
9
10 if (usuario ==maquina):
11     print(f'Opcion del Usuario {usuario} y Opcion de la Máquina {maquina} el resultado: Empate')
12 elif ((usuario == 1 and maquina == 3) or (usuario == 2 and maquina == 1) or (usuario == 3 and maquina == 2) ):
13     print(f'Opcion del Usuario {usuario} y Opcion de la Máquina {maquina} el resultado: Gana el usuario')
14 else:
15     print(f'Opcion del Usuario {usuario} y Opcion de la Máquina {maquina} el resultado: Gana la maquina')
16
```

Ln: 4, Col: 14

¿Seria posible pensar en usar **Estructuras de datos**?

¿Seria posible controlar que el ingreso del usuario sea correcto: solicitar una entrada hasta que ingrese una correcta?

¿Que **Estructuras de flujo de control** serviría para eso?

¿Seria posible tener varias jugadas hasta que el usuario decida no jugar mas?

¿Es posible **modularizar**?

*Fases en la **resolución de un problema***

a ser resuelto por un agente de procesamiento de información

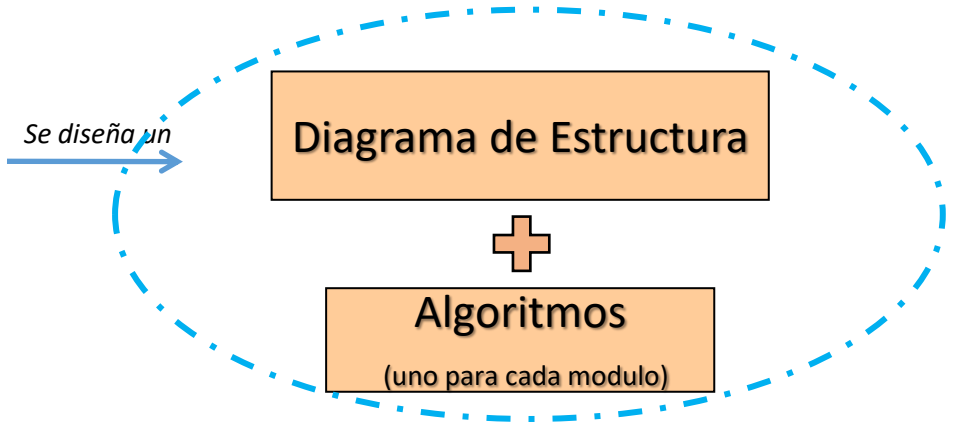
1. Definición del problema
2. Análisis del problema

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

3. Diseño del algoritmo
4. Prueba del algoritmo

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

5. Codificación del algoritmo en un lenguaje
6. Prueba y puesta a punto del programa
7. Documentación del programa



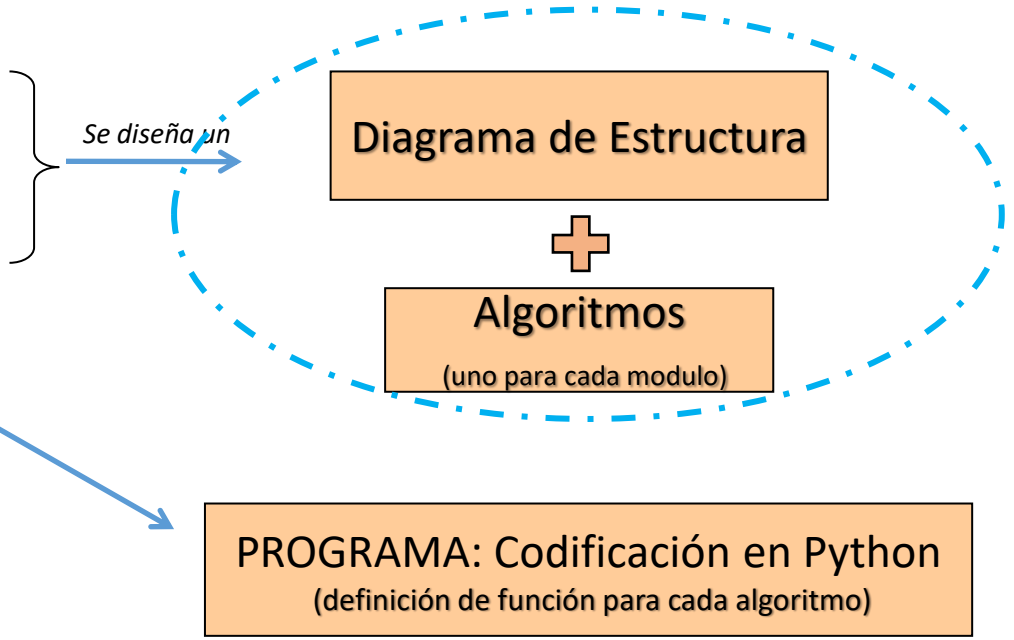
*Fases en la **resolución de un problema***

a ser resuelto por un agente de procesamiento de información

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Prueba del algoritmo
5. **Codificación del algoritmo en un lenguaje**
6. Prueba y puesta a punto del programa
7. Documentación del programa

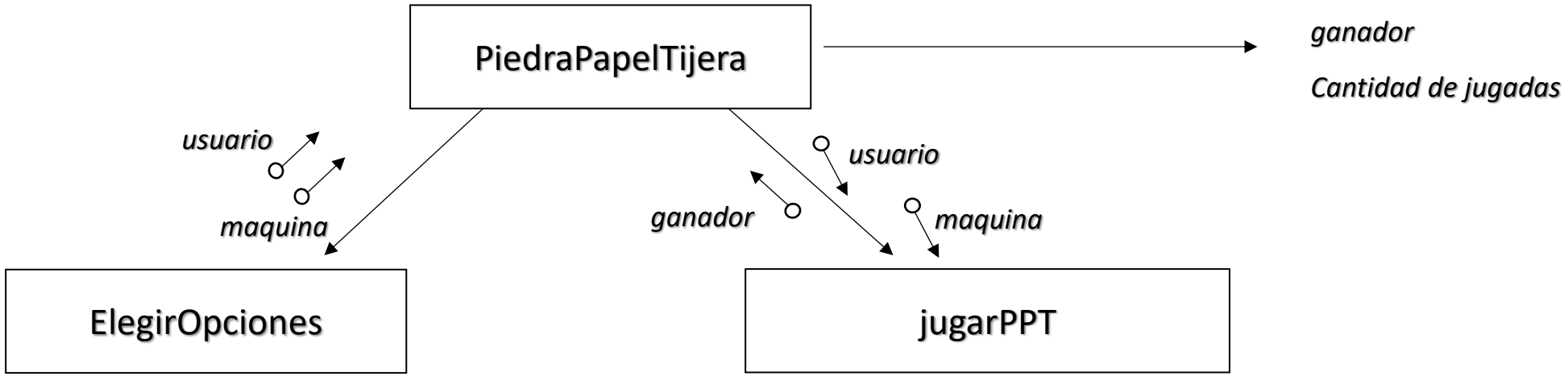
Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?



Juego: Piedra-Papel-Tijera

Diagrama de Estructura



Codificación

Juego: Piedra-Papel-Tijera

```python
import random
def ElegirOpciones():
    m = random.randint(1,3)
    #1:piedra 2:papel 3:tijera
    u = 0
    while ((u < 1) or (u > 3)):
        u=int(input('Ingrese su opcion: [1:piedra | 2:papel | 3:tijera] '))
    return(m,u)

def JugarPPT(maq,us):
    if ((us == 1 and maq == 3) or (us == 2 and maq == 1) or (us == 3 and maq == 2)):
        resultado = 'GANA USUARIO'
    else:
        resultado = 'GANA MAQUINA'
    return(resultado)


maquina,usuario = ElegirOpciones()
contar = 0
while (usuario != maquina):
    ganador = JugarPPT(maquina,usuario)
    print(f'Con la opcion del Usuario {usuario} y la opcion de la maquina {maquina} el resultado es: {ganador}')
    contar = contar + 1

    maquina,usuario = ElegirOpciones()

print(f'Con la opcion del Uuario {usuario} y la opcion de la maquina {maquina} el resultado es: EMPATE luego de {contar} jugada/s')
```