ucm

Ingeniería Ejecución en Computación e informática

Programación Avanzada

Clase 2: Niveles de lenguaje y nivelación de Python 1

Mg. Francisco Philip Vásquez Iglesias

Resultados de aprendizaje

RESULTADOS DE APRENDIZAJES
1.- Describir los paradigmas de programación vigentes y como estos orientan la resolución de un problema.
2.- Analizar las características de la programación orientada a objetos determinando como estas mejoran la calidad del proceso de programación, basándose en situaciones cercanas a la comunidad.
3.- Resolver problemas identificando los objetos principales, describiendo sus atributos y comportamiento, presentando resultados en un lenguaje de acuerdo al contexto profesional.

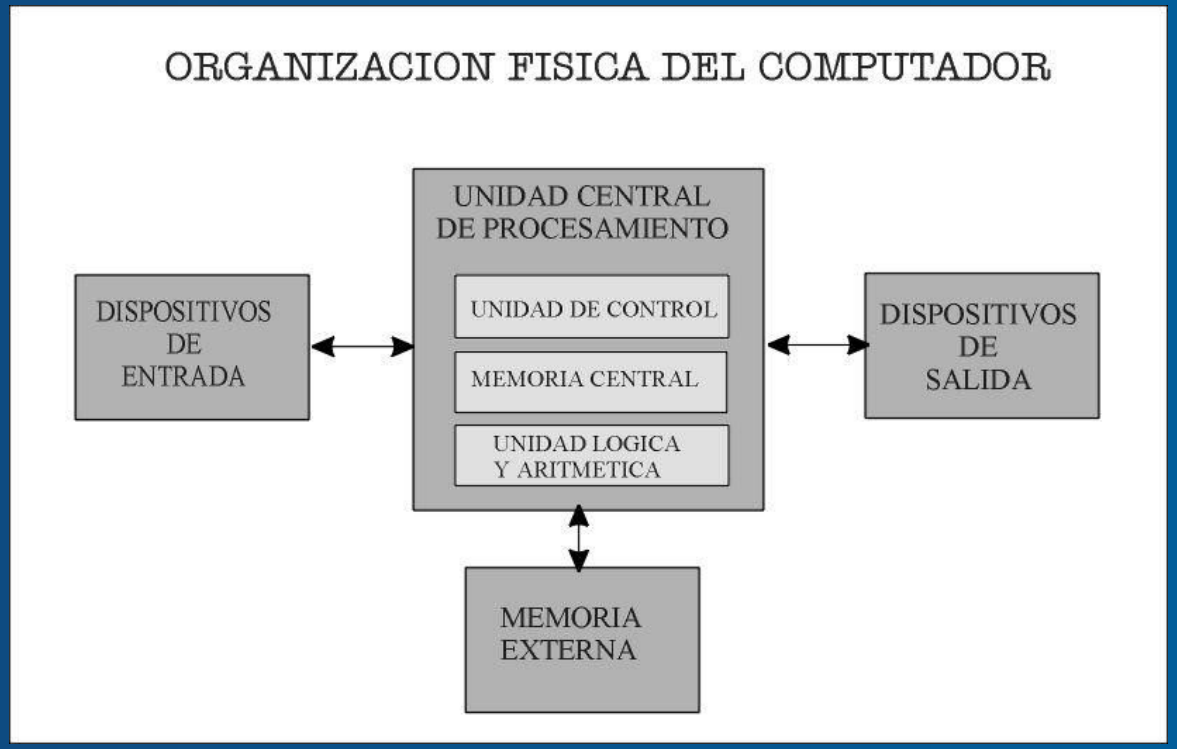
Introducción

Introducción

- Los computadores son herramientas esenciales en casi todos los campos de nuestras vidas.
- Sin una lista de instrucciones a seguir, la computadora es prácticamente inútil.
- Los lenguajes de programación nos permiten escribir programas y comunicarnos con las computadoras.

Diferencia entre datos e información

- Los datos manifiestan hechos en bruto.
- La información son datos procesados, generando datos significativos.
- Una computadora procesa datos y los convierte en información significativa.



Recordar que…

- El programa es la fuerza conductora de cualquier tarea que hace una computadora.
- Un programa es una lista de instrucciones detalladas que indican a la computadora lo que debe hacer.
- El trabajo del programador es escribir programas que influyen en la computadora, tomar datos en bruto y transformarlos en información significativa para el usuario final.

Lenguajes de programación

- Sirven para escribir programas que permitan la comunicación usuario/máquina.
- Los programas que realizan tareas concretas: nóminas, contabilidad, análisis estadístico, etc., se denominan programas de aplicación.

Tipos de lenguaje

- Los principales tipos de lenguaje utilizados en la bibliografía son:
  - Lenguajes máquina.
  - Lenguaje de bajo nivel (ensamblador).
  - Lenguaje de alto nivel.

Instrucciones a la computadora

- Normalmente el término ***instrucción*** se refiere a los lenguajes máquina y bajo nivel, mientras que sentencia o proposición para los lenguajes de alto nivel.
- Un programa consiste en una secuencia de instrucciones, cada una de ellas especifica ciertas operaciones que debe ejecutar la computadora.

Instrucciones básicas de todos los lenguajes de programación

- ***Instrucciones de entrada/salida:*** transfieren información y datos entre dispositivos periféricos y la memoria principal.
- ***Instrucciones aritmético-lógicas:*** ejecutan operaciones aritméticas (suma, multiplicación, división, etc.) y lógicas (AND, OR, NOT, etc.).
- ***Instrucciones selectivas:*** permiten la selección de tareas alternativas en función de los resultados de diferentes expresiones condicionales.
- ***Instrucciones repetitivas:*** permiten la repetición de secuencias de instrucciones un número determinado de veces.

Lenguaje Máquina

- Son aquellos que están escritos en lenguajes directamente entendibles por la máquina.
- Sus instrucciones son cadenas binarias que especifican una operación.
- Las direcciones de memoria implicadas en la operación se denominan instrucciones de máquina o código máquina.
- Las instrucciones en lenguaje máquina dependen del hardware de la computadores, por lo que difiera de una computadora a otra.

Ventajas/desventajas de programar en lenguaje máquina

- Ventajas:
  - Permite la posibilidad de cargar (transferir un programa a la memoria) sin necesidad de traducción posterior.
  - Velocidad de ejecución superior a cualquier otro lenguaje de programación.
- Desventajas:
  - Dificultad y lentitud en la codificación.
  - Poca fiabilidad.
  - Dificultad grande de verificar y alistar los programas.
  - Los programas sólo son ejecutables en el mismo procesador.

Conclusión

- Para evitar que los usuarios trabajen sobre los lenguajes máquina, se crearon otros lenguajes que permiten escribir programas con instrucciones similares al lenguaje humano.
- Estos lenguajes se denominan de alto y bajo nivel.

Lenguajes de bajo nivel

- Son más fáciles de utilizar que los lenguajes de máquina, aunque también dependen de la máquina en particular.
- El lenguaje de bajo nivel por excelencia es el ensamblador.
- Las instrucciones lenguaje ensamblador son conocidas como instrucciones mnemotécnicas, debido que buscan una fácil memorización. Por ejemplo ADD, SUB, DIV, etc.

Ejemplo en Emu8086

Observaciones

- Un programa escrito en lenguaje ensamblador no puede ser ejecutado directamente por la computadora, sino que requiere una fase de traducción al lenguaje máquina.
- El programa original escrito en lenguaje ensamblador se denomina programa fuente y el programa traducido en lenguaje máquina se conoce como programa objeto.
- El traductor de programas fuente a objeto se llama ensamblador.

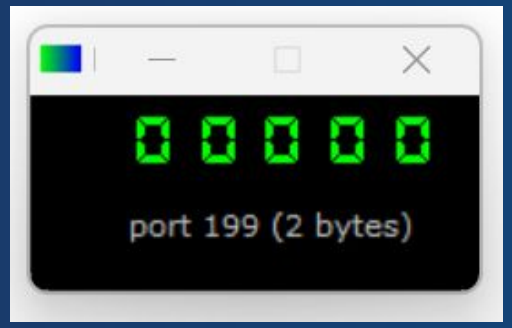
Programa ensamblador

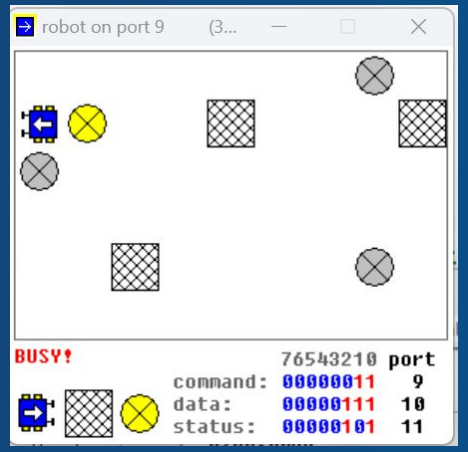
Ventajas/desventajas de programar en ensamblador

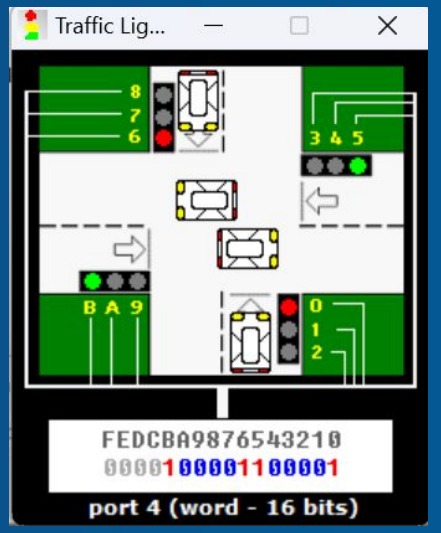
- Ventajas frente a los lenguajes máquina:
  - Mayor facilidad de codificación.
  - En general su velocidad de cálculo es mayor.
- Desventajas:
  - Dependencia total de la máquina, lo que impide la transportabilidad de los programas.
  - El lenguaje ensamblador del PC es distinto del lenguaje ensamblador del Apple Macintosh o otras compañías.
  - Las formación de los programadores es más compleja a la de los programadores alto nivel.
  - Exige conocimiento del interior de la máquina.

Uso actual de los lenguajes ensambladores

- Se centran en aplicaciones de tiempo real, control de procesos y de dispositivos electrónicos.
- Usos muy reducidos en la programación de aplicaciones.







Lenguajes de alto nivel

- Son los lenguajes más utilizados por los programadores.
- Están diseñados para que las personas escriban y entiendan los programas de un modo mucho más fácil que los lenguajes máquina y ensambladores.
- Las instrucciones del programa no dependen del diseño del hardware o de una computadora en particular (programas transportables).

Ventajas de los lenguajes de alto nivel

- Ventajas:
  - El tiempo de formación de los programadores es relativamente corto comparado con otros lenguajes.
  - La escritura de programas se basa en reglas sintácticas similares a los lenguajes humanos.
  - Las modificaciones y preparaciones de los programas son más fáciles.
  - Reducción del coste de los programas.
  - Transportabilidad.

Desventajas de los lenguajes de alto nivel

- Desventajas:
  - Incremento del tiempo de preparación, al necesitar diferentes traducciones del programa fuente para conseguir el programa definitivo.
  - No se aprovechan los recursos internos de la máquina.
  - Aumento en la ocupación de memoria.
  - El tiempo de ejecución de los programas es mucho mayor.

Lenguajes de alto nivel más utilizados en bibliografía del curso (2003)

- ***Más utilizados:***
  - C, C++, COBOL, FORTRAN, Pascal, Visual BASIC, Java y C#.
- ***En extensión:***
  - Ada-95, Modula-2, Prolog, LISP, Smalltalk y Eiffel.
- ***De gran uso en el mundo profesional:***
  - Borland Delphi, SQL y Power Builder.
- ***Para internet:***
  - Java, HTML, XML, JavaScript, Visual J, C# y PHP.

Oferta de puestos de trabajo por lenguaje (“Most Demanded Programming Languages”)

- Ranking basado en ofertas de empleo de los principales portales, incluye 7 millones de ofertas entre octubre 2021 y junio 2022.
  1. JavaScript/TypeScript.
  2. Python.
  3. Java.
  4. C#.
  5. PHP.
  6. C++.
  7. Ruby.
  8. Go.
  9. SQL.
  10. Scala.

Popularidad de los distintos lenguajes (“TIOBE Index for February 2023”)

- Ranking basado en portales de búsqueda, número de desarrolladores formados en ellos, números de cursos disponibles y empresas que los utilizan.
  1. Python.
  2. C.
  3. C++.
  4. Java.
  5. C#.
  6. Visual Basic.
  7. JavaScript.
  8. SQL.
  9. Assembly language.
  10. PHP.

Popularidad de los distintos lenguajes (“TIOBE Index for July 2024”)

- Ranking basado en portales de búsqueda, número de desarrolladores formados en ellos, números de cursos disponibles y empresas que los utilizan.
  1. Python.
  2. C++.
  3. C.
  4. Java.
  5. C#.
  6. JavaScript.
  7. Go.
  8. Visual Basic.
  9. Fortran.
  10. SQL.
  11. Delphi/Object Pascal
  12. Matlab.
  13. Rust.
  14. Ruby
  15. Scratch
  16. PHP
  17. Swift
  18. Assembly language.
  19. COBOL.
  20. Kotlin.

Utilización por parte de desarrolladores (“2022 Developer Survey”)

- Ranking basado en encuesta realizada a más de 50.000 desarrolladores, preguntando el lenguaje que utilizan en sus desarrollos.
  1. JavaScript
  2. HTML/CSS
  3. SQL
  4. Python
  5. TypeScript
  6. Java
  7. C#
  8. Bash/Shell
  9. PHP
  10. C++

Top resumen en base a lo anterior (p.1)

- ***JavaScript/TypeScript:*** Desarrollo de aplicaciones web.
- ***Python:*** Analítica y procesamiento de datos.
- ***Java:*** Desarrollo de aplicaciones multiplataforma.
- ***C#:*** aplicaciones móviles, videojuegos, aplicaciones de escritorio, backend y frontend de aplicaciones web.
- ***C++:*** bases de datos, navegadores web, sistemas operativos, compiladores y videojuegos.

Top resumen en base a lo anterior (p.2)

- C: Compiladores, intérpretes, editores, sistemas operativos y programación embebida.
- PHP: aplicaciones de escritorio y scripts en aplicaciones de arquitectura cliente-servidor.
- SQL: bases de datos.
- Ruby: Desarrollo de aplicaciones web.
- Visual Basic: automatización de tareas y aplicaciones de escritorio.

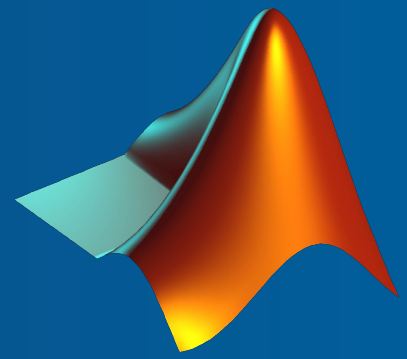
Traductores de lenguaje

Traductores de lenguaje

- Son programas que traducen a su vez los programas fuente escritos en lenguajes de alto nivel a código máquina.
- Los traductores se dividen en:
  - Intérpretes.
  - Compiladores.

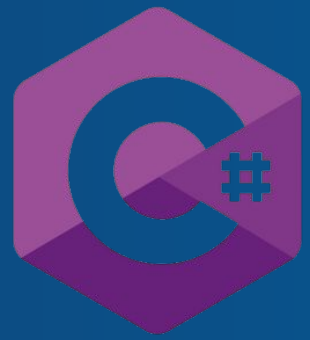
Intérpretes

- Es un traductor que toma un programa fuente, lo traduce, y a continuación lo ejecuta.
- La traducción es realizada línea por línea:
- Conocidos lenguajes interpretados son: Java, Python y Matlab.
- Algunas ventajas:
  - Tipos dinámicos.
  - Tamaño de programa más pequeños.
  - Facilidad de depuración.
- Principal desventaja:
  - Velocidad de ejecución mucho más lenta.



Compiladores

- Realizan la traducción de un programa escrito en un lenguaje de alto nivel a un lenguaje que puede entender la computadora: un lenguaje de bajo nivel.
- Antes de ejecutar un programa en un lenguaje de alto nivel, se debe ejecutar el compilador en el programa.
- A continuación, solo se necesitará una pequeña traducción adicional automática para terminar la traducción a lenguaje máquina.



Material de nivelación

Strings en Python

Declaración de String

```
miString1= 'Hola soy Philip'
print(miString1)
print("Mi String es: ",miString1)
```

```
Hola soy Philip
Mi String es:  Hola soy Philip
>>>
```

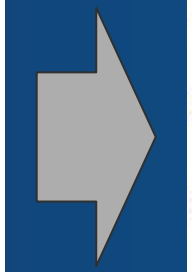
Concatenación de String

```
miString1= 'Hola soy Philip'
print(miString1)

#\n: Indica salto de línea, no se muestra
miString2= ", ¿Cómo están?\n"
print(miString2)

print("Mi String es: ",miString1+miString2)

miString3= miString1+miString2
print("Otra forma de mostrarlo: ",miString3)
```



```
Hola soy Philip
, ¿Cómo están?

Mi String es:  Hola soy Philip, ¿Cómo están?

Otra forma de mostrarlo:  Hola soy Philip, ¿Cómo están?

>>>
```

Contar apariciones

```
miString1= 'Hola soy Philip'
#\n: Indica salto de línea, no se muestra
miString2= ", ¿Cómo están?"

miString3= miString1+miString2
print("Otra forma de mostrarlo: ",miString3)

numeroDeO= miString3.count('o')
print("El número de caracteres 'o' es: ",numeroDeO)
```

```
Otra forma de mostrarlo:  Hola soy Philip, ¿Cómo están?
El número de caracteres 'o' es:  3
>>>
```

Contar apariciones de cadenas

```
miString1= 'Hola soy Philip'
#\n: Indica salto de línea, no se muestra
miString2= ", ¿Cómo están?"

miString3= miString1+miString2+" Philip"
print("Otra forma de mostrarlo: ",miString3)

apariciones= miString3.count('Philip')
print("El número de 'Philip' es: ",apariciones)
```

```
Otra forma de mostrarlo:  Hola soy Philip, ¿Cómo están? Philip
El número de caracteres 'o' es:  3
El número de 'Philip' es:  2
>>>
```

Buscar en subcadenas

```
miString1= 'Hola soy Philip'
#\n: Indica salto de línea, no se muestra
miString2= ", ¿Cómo están?"

miString3= miString1+miString2+" Philip"
print(miString3,"\n")

print(miString3[0:10])
apariciones= miString3.count('soy',0,10)
print("El número de apariciones es: ",apariciones)

print(miString3[10:20])
apariciones= miString3.count('soy',10,20)
print("El número de apariciones es: ",apariciones)
```

```
Hola soy Philip, ¿Cómo están? Philip

Hola soy P
El número de apariciones es:  1
hilip, ¿Có
El número de apariciones es:  0
>>>
```

Cambiar a mayúscula el primer carácter de una cadena

```
print("Ingrese su nombre:")
nombre=input()
print("Su nombre es: ",nombre)
nombre=nombre.capitalize()
print("Su nombre es: ",nombre)
```

```
Ingrese su nombre:
philip
Su nombre es:  philip
Su nombre es:  Philip
>>>
```

**Si la cadena posee más de una palabra, el método se aplica a cada palabra.*

Separar cadenas

```
cadena= "tecito y pan"
cadenaSeparado= cadena.split(' ')
print(cadena)
print(cadenaSeparado)

cadena= "tecito,pan,café"
cadenaSeparado= cadena.split(',')
print(cadena)
print(cadenaSeparado)
```

```
tecito y pan
['tecito', 'y', 'pan']
tecito,pan,café
['tecito', 'pan', 'café']
>>>
```

Buscar una cadena

```
cadena= "paralelelogramo paralelepipedo"
ubicación= cadena.find("le")
print(ubicación)

largo=len(cadena)
print(largo)
ubicación2= cadena.find("le",10,largo)
print(ubicación2)
```

```
4
30
20
>>>
```

**También existe index() que envía ValueError si no encuentra nada.*

Cadena el minúscula o mayúscula

```
miCadena1= "philip"
miCadena2= "PHILIP"
miCadena3= "Philip"
print(miCadena1.islower())
print(miCadena2.isupper())
print(miCadena3.islower())
print(miCadena3.isupper())
```

```
True
True
False
False
>>>
```

Transformar cadena a minúscula o mayúscula

```
miCadena= "Philip"
print(miCadena.lower())
print(miCadena.upper())
```

```
philip
PHILIP
>>>
```

Cantidad de “a” y “e” en las cadenas

```
#Determinar cuántos caracteres a y e posee cada cadena
a=cadena1.count("a")
e=cadena1.count("e")
total1=a+e
print("La primera cadena posee", total1, "caracteres 'a' y 'e'")

a=cadena2.count("a")
e=cadena2.count("e")
total2=a+e
print("La primera cadena posee", total2, "caracteres 'a' y 'e'")

a=cadena3.count("a")
e=cadena3.count("e")
total3=a+e
print("La primera cadena posee", total3, "caracteres 'a' y 'e'")
```

```
La primera cadena posee 12 caracteres 'a' y 'e'
La primera cadena posee 23 caracteres 'a' y 'e'
La primera cadena posee 11 caracteres 'a' y 'e'
>>>
```

Contador de palabras con mezclas de mayúsculas y minúsculas

```
cadenaSeparada1= cadena1.split(' ')
print(cadenaSeparada1)
tama=len(cadenaSeparada1)
contador=0
for i in range(tama):
    bandera=0
    esMayus=cadenaSeparada1[i].isupper()
    esMinus=cadenaSeparada1[i].islower()
    if(esMayus==False and esMinus==False):
        contador=contador+1
print("La cadena 1 tiene",contador,"palabras que mezclan mayus y min")
```

```
['El', 'valor', 'de', 'una', 'idea', 'radica', 'en', 'el
', 'Uso', 'de', 'la', 'misma']
La cadena 1 tiene 2 palabras que mezclan mayus y min
La cadena 2 tiene 5 palabras que mezclan mayus y min
La cadena 3 tiene 3 palabras que mezclan mayus y min
>>>
```

Número de palabras que comienzan con minúscula

```
cadenaSeparada1= cadena1.split(' ')
cadenaSeparada2= cadena2.split(' ')
cadenaSeparada3= cadena3.split(' ')
tama1= len(cadenaSeparada1)
tama2= len(cadenaSeparada2)
tama3= len(cadenaSeparada3)

contador=0
for i in range(tama1):
    minus=cadenaSeparada1[i][0].islower()
    if(minus==True):
        contador=contador+1
print("La cadena 1 tiene",contador,"palabras que comienzan con min")
```

```
La cadena 1 tiene 10 palabras que comienzan con min
La cadena 2 tiene 17 palabras que comienzan con min
La cadena 3 tiene 9 palabras que comienzan con min
>>>
```

Reemplazar caracteres

```
cadena="El profe Philip"
nuevaCadena=cadena.replace("profe","estudiante")
print(cadena)
print(nuevaCadena)

cadena2="pan para papá"
print(cadena2)
nuevaCadena=cadena2.replace("pa","ma",2)
print(nuevaCadena)
```

```
El profe Philip
El estudiante Philip
pan para papá
man mara papá
>>>
```

Comienzo de una cadena

```
cadena="Paralelelogramo"
print(cadena.startswith("Paralelelo"))
print(cadena.startswith("Par"))
print(cadena.startswith("lel",5,15))
print(cadena.startswith("lel",6,15))
```

```
True
True
False
True
>>>
```

Cambiar una cadena de mayúscula a minúscula y viceversa

```
cadena="Francisco Philip Vásquez Iglesias"
cadenaModificada=cadena.swapcase()
print(cadena)
print(cadenaModificada)
```

```
Francisco Philip Vásquez Iglesias
fRANCISCO pHILIP vÁSQUEZ iGLESIAS
>>>
```

Ejercicios

Ejercicio 1

- Escribir un programa que pida por teclado dos palabras, y a continuación, indique si riman o no.
  - Si coinciden las tres últimas letras debe indicar que si riman.
  - Si coinciden sólo las dos últimas letras debe indicar que riman un poco.
  - En caso contrario, debe indicar que no riman.

Ejercicio 2

- Un mensaje ha llegado al mail con errores, diseñe un programa que sea capaz de corregir estos correos defectuosos, donde el sistema ha reemplazado erróneamente los espacios por comas y las comas por ‘#’.
- Un ejemplo de mensaje erróneo es: “La,cantimplora,se,calentó#tomar,agua,así,no,tiene,gracia”

Ejercicio 3

- Realice un programa que lea tantas cadenas de String como el usuario indique por teclado, a continuación, para cada cadena separe cada palabra por espacio (“ ”) y muestre en pantalla dos mensajes:
  - El primero contiene todas las palabras que ocupan una posición impar en el texto.
  - El segundo contiene todas las palabras que ocupan una posición par en el texto.
- Ejemplo: “Hugo y Philip somos sus profes”
  - Mensaje 1: “Hugo Philip sus”
  - Mensaje 2: “y somos profes”

Ejercicio 4

- Cree un programa donde sea posible ingresar una página de diario de vida, el programa deberá comprobar que el escrito cumpla con los siguientes estándares mínimos:
  - La primera palabra del escrito debe comenzar con mayúscula.
  - La primera palabra luego de un punto seguido deberá comenzar con mayúscula.

Ejercicio 5

- Cree un programa donde se ingrese una cadena de String y entregue el porcentaje de letras vocales que posee, del total de vocales entregue el porcentaje de aparición de cada vocal y muestra cual es la vocal con mayor frecuencia de la cadena.

Ejercicio 6

- Realice un programa que solicite al usuario una cadena String y, a continuación, solicite al usuario un recorte de cadena en un intervalo específico (desde la posición 1 hasta el largo del String).
- Por ejemplo, si se ingresa “*Philip quiere que aprendan mucho*”, con el intervalo 5 a 15, el resultado será: “ip quiere “

*Acomode el intervalo de valores, ya que Python comienza las posiciones desde el 0.

Ejercicio 7

- Realizar un programa donde se solicite una cadena al usuario, a continuación, el programa imprimirá la cadena invertida caracter a caracter y una cadena invertida en grupos de 4 caracteres.
- Ejemplo:
  - Entrada: “Ingeniería en la UCM”
  - Salida: “ MCU al ne aíreinegnI”
    “ UCMn laía enierInge”

Ejercicio 1: Pangramas

- Textos como "The quick brown fox jumps over the lazy dog" o "El veloz murciélago hindú comía feliz cardillo y kiwi. La cigüeña tocaba el saxofón detrás del palenque de paja". Son "pangramas", textos que contienen todas las letras de un cierto alfabeto, quizá repetidas.
- Crear un programa que reciba varias frases y que diga si cada una de ellas es un pangrama o no. Sólo deberás considerar las letras del alfabeto inglés (no considerar las vocales acentuadas ni la eñe, entre otras), que podrán aparecer en mayúsculas o en minúsculas.
- Para cada frase deberás responder SI cuando se trate de un pangrama o NO cuando no lo sea.

Ejercicio 8: Calculadora de ecuaciones cuadráticas

- Realice un programa en python, tal que sea capaz de resolver ecuaciones cuadráticas con solución, para esto se debe solicitar al usuario los coeficientes a, b y c.
- Recordar que la ecuación para resolver funciones cuadráticas es como sigue:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Ejercicio 9: Año bisiesto

- Realizar un algoritmo que, dado un año, este indique si es un año bisiesto o no. Las condiciones son las siguientes:
  - Si es divisible por 4 y no es divisible 100 es bisiesto.
  - Si un año es divisible entre 100 y además es divisible entre 400, también resulta bisiesto.

Ejercicio 10: Número menor que 1000

- Hacer un algoritmo que lea un número menor que 1000:
  - Si el número tiene un dígito, deberá elevarlo al cuadrado y mostrar su resultado.
  - Si el número es de dos dígitos, multiplicarlo por dos y mostrar su resultado.
  - Si el número es de tres dígitos, restarle cien y mostrar el resultado.
  - En otro caso, mostrar la leyenda “Número no válido”.

Ejercicio 11:

- Leer un archivo de texto con un conjunto de líneas de texto, a continuación, indicar por pantalla la cantidad de palabras que comienzan con vocales y la cantidad de palabras de terminan con consonantes.

ucm

Ingeniería Ejecución en Computación e informática

Programación Avanzada

Fin de la clase

Mg. Francisco Philip Vásquez Iglesias