Club de Datos

Identidad

Presidencia

nuevo CECEN

Pensamiento computacional

¿Qué es?

Pensamiento computacional

¿Qué es?

"*[El Pensamiento computacional]…implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, basándose en los conceptos fundamentales de la ciencia de la computación. El pensamiento computacional incluye una amplia variedad de herramientas mentales que reflejan la amplitud del campo de la computación…[además] representa una actitud y unas habilidades universales que todos los individuos, no sólo los científicos computacionales, deberían aprender y usar*"

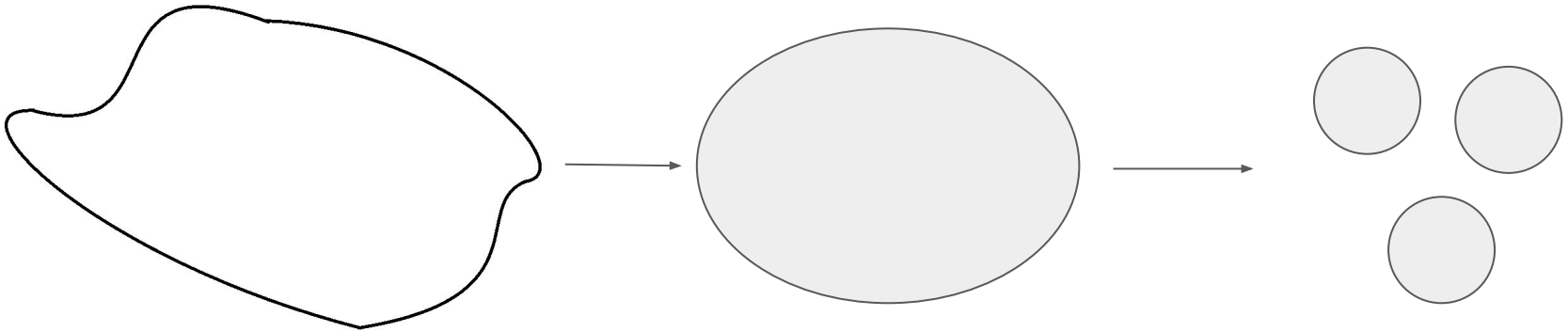
Jeannette Wing

Pensamiento computacional

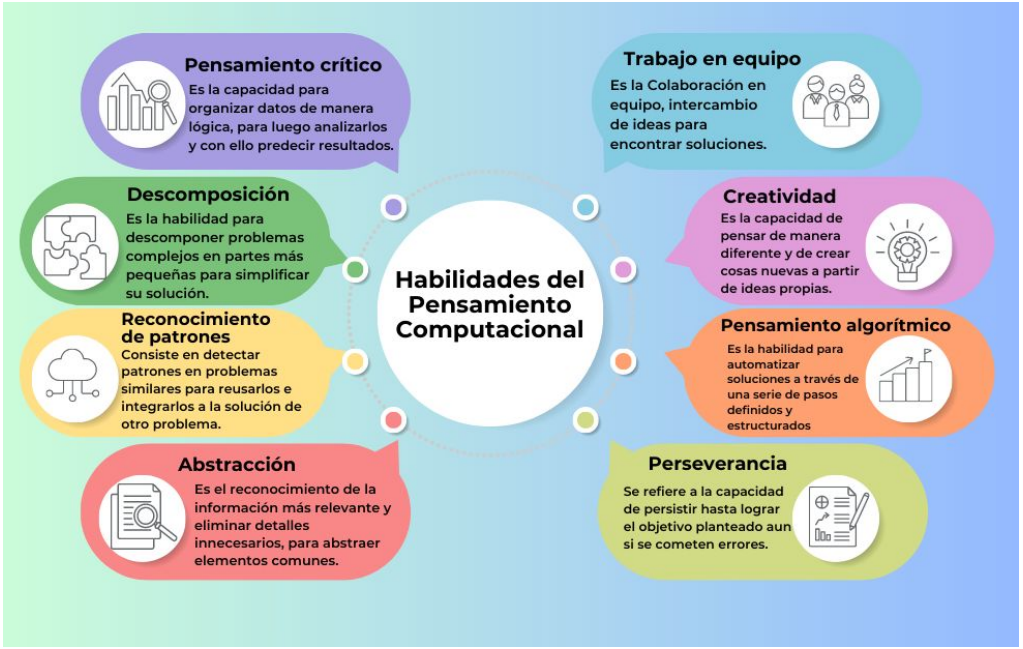
¿Qué es?

"*[El Pensamiento computacional]…implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, basándose en los conceptos fundamentales de la ciencia de la computación. El pensamiento computacional incluye una amplia variedad de herramientas mentales que reflejan la amplitud del campo de la computación…[además] representa una actitud y unas habilidades universales que todos los individuos, no sólo los científicos computacionales, deberían aprender y usar*"

Jeannette Wing



¿Cómo pensamos computacionalmente?



Un poco de código

```
variable1 = 10
variable2 = 20
variable3 = "Club de datos"
```

```
variable4 = variable1 + variable2 # ¿Qué es?
```

```
print(variable4)
```

```
30
```

pero, ¿Qué nombres les ponemos a las variables?

¿son buenos nombres de variable…?

- variable1
- variable2
- contenedor
- x

¿Por qué?

Entonces, ¿Qué nombres les ponemos a las variables?

Queremos nombrar a las variables de forma **declarativa**, que se entienda qué representan sin tener que leer el código entero para adivinarlo.

Buenos ejemplos:

- sumatoria_de_precios
- nombre_usuario
- edad_cliente
- area_figura

Entonces, ¿Qué nombres les ponemos a las variables?

Queremos nombrar a las variables de forma **declarativa**, que se entienda qué representan sin tener que leer el código entero para adivinarlo.

Buenos ejemplos:

- sumatoria_de_precios
- nombre_usuario
- edad_cliente
- area_figura

Más malos ejemplos:

- nombre_del_tercer_cliente_en_la_fila
- suma_de_los_precios_mayores_que_el_umbral
- puntos_que_forman_la_figura
- variable

Sentencias básicas

```
print(3)
print('Hola Mundo')
print(variable4)
```

```
nombre = input()
```

Codeemos

1) Hacer un programa en python que, al ejecutarse, muestre “Hola Mundo” en consola.

2) “...” preguntarle al usuario su nombre y saludarlo mostrando “Hola {input}” en consola

Tipos

el TIPO de un dato nos dice a qué dominio pertenece, qué operaciones podemos realizar con él, y cómo es representado internamente

Tipos

el TIPO de un dato nos dice a qué dominio pertenece, qué operaciones podemos realizar con él, y cómo es representado internamente

DATO	Tipo (en Python)
7	int (entero)
12.4	float (numero con coma)
“Buenos Aires”	string (cadena de texto)
True	bool
False	bool
[1,2,”Hola”,4]	list

Tipos

Charlemos (y probemos)

3 + 2.5

Tipos

Charlemos (y probemos)

3 + 2.5

3 + “hola”

```
TypeError: can only concatenate
list (not "str") to list
```

“hola” + “ que tal”

[1,2] + [3,4]

[1,2] + “3 4”

Codeemos

Hacer un programa que le pida al usuario dos números, los sume, y muestre el resultado en consola.

Estructura condicional

Queremos hacer programas que no se comporten siempre de la misma manera, por ejemplo...

Un sistema de login, que pregunte la edad:

- Si la edad es >= 18. OK
- sino, error

Estructura condicional

La mayoría de los lenguajes lo implementan con una estructura parecida a la siguiente

```
if (condición):
    c1 (código si da True)
else:
    c2 (código si da
False)
```


si (condición), entonces c1. Sino, c2

Estructura condicional

```
if (condición):
    c1 (código si da True)
else:
    c2 (código si da False)
```

¿Qué es, exactamente, condición?

Cualquier cosa que de un dato de tipo bool (booleano), es decir, algo que tenga valor True o False.

Estructura condicional

if (condición):
    c1 (código si da True)
else:
    c2 (código si da False)

¿Qué es, exactamente, condición?

Cualquier cosa que de un dato de tipo bool (booleano), es decir, algo que tenga valor True o False.

```
if x > 4:
    print("x es mayor a 4")
else:
    print("x es menor o igual a 4")
```

¿Qué se va a mostrar por consola?

Estructura condicional

coodemos

1) pedirle al usuario un número. Si el número es menor a 10 o mayor a 100, felicitarlo, sino, decirle que perdió.

2) pedir un número. si está entre 6 y 12, felicitarlo. sino, decirle que perdió.

operadores lógicos en Python

and
or
not

Pensemos el sig. problema

pedirle al usuario un número. Si el número es menor a 10, felicitarlo. Sino, **le pedimos otro** y aplicamos la **misma condición.**

Pensemos el sig. problema

pedirle al usuario un número. Si el número es menor a 10, felicitarlo. Sino, **le pedimos otro** y aplicamos la **misma condición.**

¡No podemos!
No sabemos cuántas veces vamos a tener que pedirle que ingrese un número

Necesitamos **ciclos.**

Ciclos

estructura que nos permite repetir una serie de acciones

X cantidad de veces.

```
for i in range(x):
    print(i)
```

```
for i in range(10):
    print(i)
```

0, 1, 2 ,3, 4, 5, 6, 7, 8, 9

¡Ojo los índices!

mientras una condición sea verdadera

```
while x < 10:
    print(x)
    x = x + 1
```

en este caso, x < 10 es la condición. es **necesario** que dentro del ciclo haya algo que nos *acerque* a la terminación del mismo

Ciclos

Si usamos for, sabemos cuántas iteraciones vamos a hacer.
Con while no.

pedirle al usuario un número. Si el número es menor a 10, felicitarlo. Sino, **le pedimos otro** y aplicamos la **misma condición.**

```
es_mayor = True
while (es_mayor):
  print('Ingrese un numero. ')
  entrada = int(input())
  if entrada < 10:
    print('Te felicito')
    es_mayor = False
```

Ciclos

Notar que si usamos for, sabemos en qué iteración estamos más fácilmente.

```
x = 0
for i in range(5):
  x = x + i

print(x) # Qué valor tiene x al final?
```

Ciclos

codeemos

Dado un número x, determinar si es primo.
Pista: pensarlo con el resto de la división entera

Ciclos

codeemos

Dado un número x, determinar si es primo.
Pista: pensarlo con el resto de la división entera

```
x = 10
es_primo = True
for i in range(x):
  if x % i == 0:
    es_primo = False

if es_primo:
  print("Es primo")
else:
  print("No es primo")
```

¡Recreo!

Listas

Secuencia ordenada de elementos. En python, pueden ser de tipos distintos

Listas

Secuencia ordenada de elementos. En python, pueden ser de tipos distintos

```
lista_de_numeros = [1,2,3,4]
lista_de_nombres = ["Fede","Sofi","Sol"]
lista_de_cosas = [1,"Buenos aires",[1,4],(4,2)] # No es buena idea
```

¿Qué usos se les ocurren?

Listas

Secuencia ordenada de elementos. En python, pueden ser de tipos distintos

```
lista_de_numeros = [1,2,3,4]
lista_de_nombres = ["Fede","Sofi","Sol"]
lista_de_cosas = [1,"Buenos aires",[1,4],(4,2)] # No es buena idea
```

un atributo **super** importante y usado de una lista es su **longitud**

```
lista = [2,3,4,5]
print(len(lista))
```

```
4
```

Listas

Cómo las manipulamos

Creación

```
lista1 = [] # Creación de una lista vacía
lista2 = [1,4,2] # Creación de una lista con elementos
```

Agregar un elemento

```
# Agregar un elemento al final
lista1 = []
lista1.append(4)
print(lista1)
```

[4]

Sacar un elemento

```
# eliminar el elemento de la posición i.
lista1 = ["Buenos Aires","Córdoba","Corrientes"]
lista1.pop(1)
print(lista1)
```

[“Buenos Aires”,”Corrientes”]

Listas

Acceso

```
# Accedemos con [i], siendo i el índice
# que queremos obtener.
lista = [1,2,3,4]
print(lista[0])
```

```
1
```

```
# ejemplo
lista = [1,2,3]
suma = lista[1] + lista[2]
print(suma)
```

```
5
```

Listas

Codeemos!

1) Dado una lista llamada numeros, eliminar todos los mayores a 10

2) Dado un número X, **generar** una lista con los pares menores a ese número

Funciones

Google: es un bloque de líneas de código o un conjunto de instrucciones cuya finalidad es realizar una tarea específica.

Intuición: forma de abstraer, encapsular y **reutilizar** código.

```
def funcion(parametro1, parametroN):
  # bla
  # bla
  return resultado
```

Funciones - ejemplo

```
lista = [1,4,2]
lista2 = [5,2,6]
sumatoria1 = 0
for elem in lista:
  sumatoria1 = sumatoria1 + elem
sumatoria2 = 0
for elem in lista2:
  sumatoria2 = sumatoria2 + elem
print("Sumatoria 1: ", sumatoria1)
print("Sumatoria 2: ", sumatoria2)
```

```
Sumatoria 1:  7
Sumatoria 2:  13
```

¿Qué problema ven?

```
def sumatoria(lista):
  resultado = 0
  for elem in lista:
    resultado = resultado + elem
  return resultado


lista = [1,4,2]
lista2 = [5,2,6]
sumatoria1 = sumatoria(lista)
sumatoria2 = sumatoria(lista2)
print("Sumatoria 1: ", sumatoria1)
print("Sumatoria 2: ", sumatoria2)
```

lo puedo hacer las veces que quiera sin volver a escribir código

Funciones - más ejemplos

```python
def minimo_de_una_lista(lista):
  minimo = lista[0]
  for elemento in lista:
    if elemento < minimo:
      minimo = elemento
  return minimo
```

Funciones - más ejemplos

```
def minimo_de_una_lista(lista):
  minimo = lista[0]
  for elemento in lista:
    if elemento < minimo:
      minimo = elemento
  return minimo
```

```
def lista_mas_larga(lista1,lista2):
  if len(lista1) > len(lista2):
    res = lista1
  else:
    res = lista2
  return res
```

Funciones - más ejemplos

```python
def minimo_de_una_lista(lista):
  minimo = lista[0]
  for elemento in lista:
    if elemento < minimo:
      minimo = elemento
  return minimo
```

```python
def lista_mas_larga(lista1,lista2):
  if len(lista1) > len(lista2):
    res = lista1
  else:
    res = lista2
  return res
```

```python
def funcion_matematica_rara(x,y,z,alpha):
  return ((x ** 2) (y+z))/ 1 ** - alpha
```

¿Preguntas?

¿Qué es un algoritmo?

no confundir con logaritmo

La solución **formal** a un problema.

Tiene que ser formal porque la queremos **implementar** en nuestro programa

¿Qué es un algoritmo?

no confundir con logaritmo

La solución **formal** a un problema.

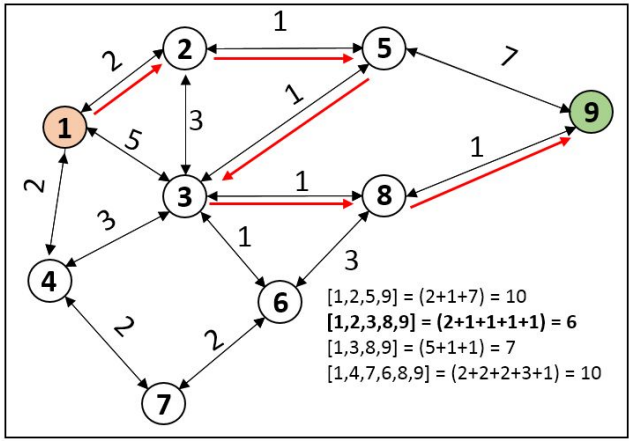
Tiene que ser formal porque la queremos **implementar** en nuestro programa

Listo.

Algoritmo - ejemplo (motivador)

Camino más corto entre dos puntos de un grafo (una red)

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```



Problema de ordenar una lista

“Sorting”

Problema de ordenar una lista

“Sorting”

- Problema importante en la computación: muchos, muchísimos, usos. Por eso queremos hacerlo bien.
- Simple: ordenar de menor a mayor.

Problema de ordenar una lista

“Sorting”

- Problema importante en la computación: muchos, muchísimos, usos. Por eso queremos hacerlo bien y súper rápido.
- Simple: ordenar de menor a mayor.

Pensemos entre todos algún algoritmo para ordenar una lista.

[7,4,5,2,1] ⟶ [1,2,4,5,7]

Alerta de spoiler

Selection sort

Input = lista_original. Output = lista_ordenada

1) Busco el mínimo de lista_original
2) Lo agrego al final de la lista_ordenada
3) Repito hasta que ví todos los elementos de la lista_original
4) Finalmente, lista_ordenada está ordenada

Impementación en python