VARIABLES DEFINICIÓN

ALGO PARA CARGAR CUALQUIER TIPO DE DATO Y PUEDE VARIAR

VALOR QUE PUEDE SER CONSTANTE COMO VARIABLE

DONDE PODEMOS PONER DATOS, CAMBIARLOS

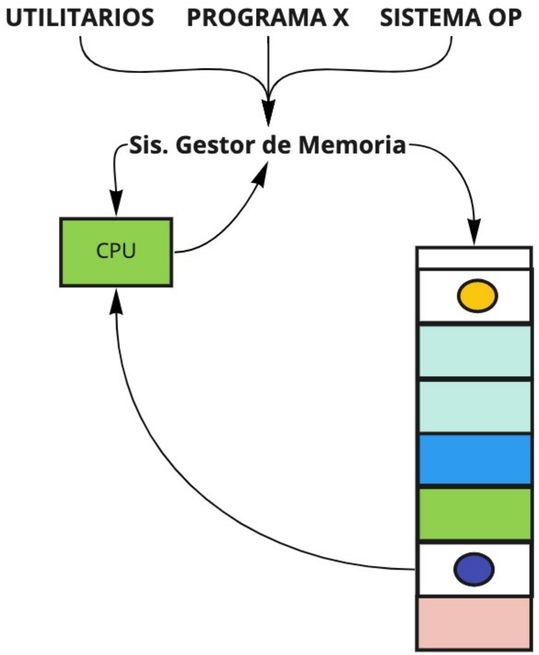
VALOR QUE PUEDE SER MODIFICADO PARA OBTENER RESULTADOS

LLEVA UNA ENTRADA Y SALIDA Y PUEDE MOSTRA EL VALOR

Es un espacio en memoria (RAM) destinado a almacenar datos

- **VALOR**
- **MODIFICABLE**
- **REUTILIZABLE**

COMPORTAMIENTO PC



Palabras reservadas

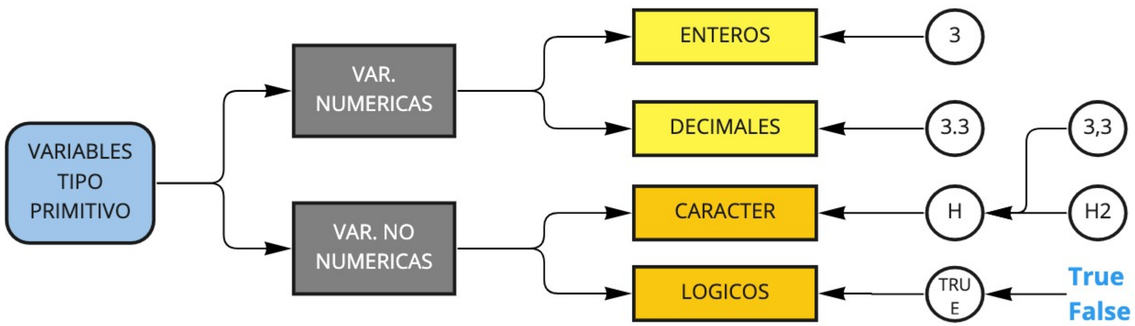
```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```

DECLARAR VARIABLES EN PYTHON

numCasa = 435

Nombre Var — **Operador** — **Valor**

ESQUEMA DE TIPOS DE DATO Y VARIABLE



SANA PRÁCTICA DEF. VARIABLES

Representativa

- **X1** —**porque despues olvidamos el propósito de la variable** ✗
- **X1 - X2 - Xn** —**porque se confunde la variable** ✗
- **sumPasajeBus** ✓

No empiezan con Números

- **2Carros** ✗
- **carros2** ✓

uso de estadares

- Camel: **_numero_casa**, **Numero_Casa**
- Pascal: **numeroCasa**, **NUMEROCASA**

Los nombre de variables no puedes ser palabras reservada ✗

inicializar variables

- **var = 0**
- **var = ' '** ✓

OPERADORES LOGICOS

AND **TRUE**
OR **FALSE**
NOT

OP1	OP2	RESULTADO
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

AND
CONJUNCIONES

OP1	OP2	RESULTADO
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

OR
DISYUNCIONES

OP1	OP2	RESULTADO	NEGADO
TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE
FALSE	FALSE	FALSE	TRUE

OR —— NOT

OP1	OP2	RESULTADO	NEGADO
TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	TRUE

AND —— NOT

Tablas de verdad

IF - ELIF - ELSE

Son decisiones basadas en condiciones que se deben tomar a nivel código.

Esas desiciones se aplican con operadores lógicos.

```
if op1 == 2:
  print ('Hola')
```

```
if op1 == 2 and op2 < 5:
  print ('Hola')
```
op1 = 2
op2 = 7

```
if op1 == 2 and op2 < 5:
  print ('Hola')
```
op1 = 2
op2 = 5

```
if op1 == 2 and op2 < 5:
  print ('Hola')
```
op1 = 2
op2 = 3
Hola

```
if op1 == 2 or op2 < 5:
  print ('Hola')
```
op1 = 2
op2 = 7
Hola

```
if op1 == 2 or op2 < 5:
  print ('Hola')
```
op1 = 2
op2 = 5
Hola

```
if op1 == 2 or op2 < 5:
  print ('Hola')
```
op1 = 2
op2 = 3
Hola

```
if not (op1 == 2 or op2 < 5):
  print ('Hola')
```
op1 = 2
op2 = 3

```
if not (op1 == 2) or op2 < 5:
  print ('Hola')
```
op1 = 2
op2 = 3
Hola

```
if op1 == 2 or op2 < 5:
  ~~print ('Hola')~~
else:
  print ('Adios')
```
op1 = 1
op2 = 5
Adios

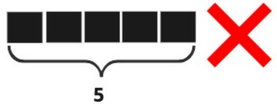
```
if op1 == 2 or op2 < 5:
  ~~print ('Hola')~~
elif  op1 < 3 and op2 <= 5
  print ('aca estoy')
else:
  print ('Adios')
```
op1 = 1
op2 = 5
aca estoy

```
if op1 == 2 or op2 < 5:
  ~~print ('Hola')~~
elif  op1 < 3 and op2 <= 5
  ~~print ('aca estoy')~~
else:
  print ('Adios')
```
op1 = 7
op2 = 5
Adios

CICLOS

while (mientras)

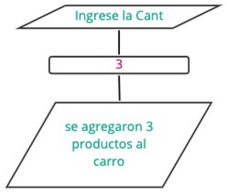
Se usa cuando no se conoce cuantas iteracciones se van a ejecutar.




```
while condicionLogica :
  print (....)
```
____ mientras se cumpla la condición haga

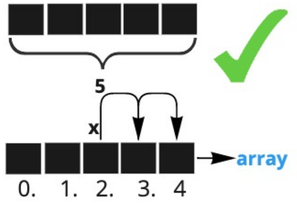
```
Productos = 10
cant = 0
while productos > 0 :
  print ('ingrese cantidad')
  cant = int(input())
  if cant > 1:
    print ('se agregaron', cant, 'productos al carro')
    productos = productos - cant
  else:
    print ('No hay suficientes productos')
```

Producto	cant
~~10~~	~~0~~
~~7~~	~~3~~
~~3~~	~~5~~
0	3



for (para)

Se usa cuando se conoce cuantas iteracciones se van a ejecutar.



for x in range (0, 5)

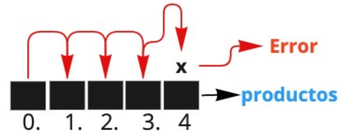
x - Es variable temporal, que solo ejecuta cuando el for esta corriendo

x - Se comporta como un iterator.

```
for x in [1, 2, 3, 4]
```

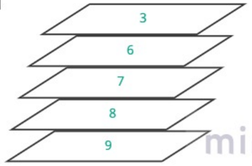
```
for x in range(5)
```

```
for x in range(0, 5, 2)
```

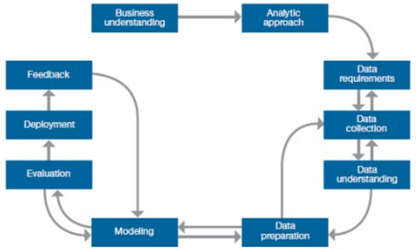



```
for x in range [0,5]:
  print (productos(x))
```

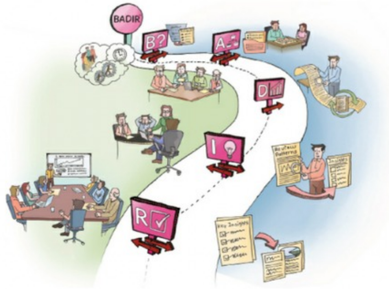
Producto	x
3	0
6	1
7	2
8	3
9	4
0	5



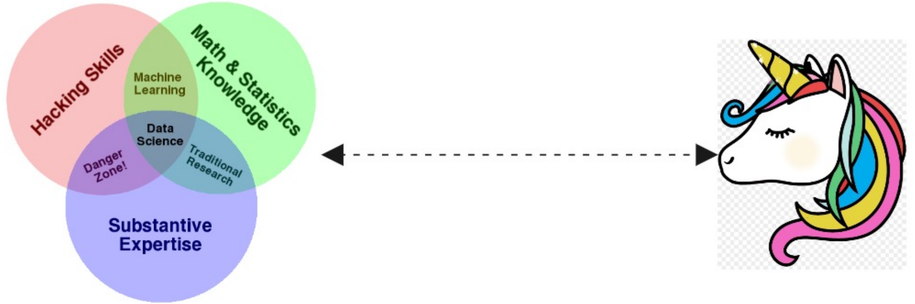
Cross Industry Standard Process for Data Mining



IBM Data Science Methodology



BADIR



METODOLOGIAS PARA LOS PROYECTOS DE CIECIA DE DATOS

Pasos a seguir de manera semi estructurada

Permiten la revision y replanteamiento para buscar el mejor resultado

Levantamiento de requerimientos

LOS DATOS NO HABLAN, PERO SI RESPONDEN A PREGUNTAS.

Logremos identificar la pregunta del negocio a responder

ENTENDER EL PROCESO DE NEGOCIO → EL SET DE DATOS NECESARIO → TRABAJAR CON LOS DATOS → IMPLEMENTAR ML → PEM

- SI YO NO SE ALGO
- LO PREGUNTO
- **NO SE ASUME**

HOMOLOGACION DE CRITERIOS

- Carga de datos.
- Trabajar con dts faltantes o perdidos.
- Estandarizar.
- Normalizar.
- Analitica.

Preparar los dts para implementar algoritmos de ML

Dividir los Set Dts 80/20.
Seleccionar el ML
Verificar su rendimiento

Si el rendimiento no optimo se ajusta el algoritmo de ML

SOBREAJUSTADOS - OVERFITING

100% ✗

DECISION SCIENCE