UT1. INTRODUCCIÓN A LOS ELEMENTOS DE UN PROGRAMA INFORMÁTICO

COMENTARIOS

- **General**
  - Los comentarios se introducen en los programas como ayuda para quienes tengan que leer dicho código, con el fin de facilitar su lectura.
  - No se ejecutan ni compilan junto con el resto del programa.
- **Python 3**
  - Comienzan con el carácter # y finalizan cuando acaba la línea

    ```
    #Esto es un comentario
    ```

  - https://docs.python.org/3/reference/lexical_analysis.html

PALABRAS RESERVADAS

- **General**
  - Son palabras que tienen que un significado especial en el lenguaje de programación.
  - No pueden ser utilizados como nombres para variables constantes, funciones, clases, etc. por el programador.
- **Python 3**
  - Palabras clave

```
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
```

  - Cierta clases de identificadores que tienen un significado especial, por ejemplo nombres definidos por el sistema.
  - https://docs.python.org/3/reference/lexical_analysis.html#reserved-classes-of-identifiers

TIPOS DE DATOS

- **General**
  - Un dato es un símbolo físico que se utiliza para representar la información.
  - Tipos de datos:
    - Numéricos: Enteros, Reales
    - Alfanuméricos: Caracteres, Cadenas
    - Booleanos (Lógicos)
    - Listas
    - Tablas
    - Referencias, apuntadores
    - …

- **Python 3**
  - All data in a Python program is represented by objects or by relations between objects.
  - None -> significa nada
  - Números:
    - Enteros (int) 32
    - Número en coma flotante (con parte decimal) (float) 4.7
    - Complejos (complex) 4.5 + 5.4j
  - Lógicos o Booleanos
    - En realidad son un tipo de número:
    - True y False / 0 y 1
  - Strings “hola”
  - Listas [4,5,6,7]
  - Tuplas 3,4,5
  - …
  - https://docs.python.org/3/reference/datamodel.html#objects-values-and-types

IDENTIFICADORES

- **General**
  - Es un nombre que identifica a una variable, a un método o función, a una clase..
  - Todos los lenguajes tienen ciertas reglas para componer los identificadores.
  - Los nombres de los identificadores deben ser significativos.
  - No pueden utilizarse como identificadores palabras reservadas.
- **Python 3**
  - Deben comenzar con una letra (a - z, A - B) o un guión bajo (_).
  - El resto de los caracteres pueden ser letras, números o guión bajo (_).
  - Distinguen entre mayúsculas y minúsculas (case sensitive).
  - Pueden tener cualquier longitud.
  - https://docs.python.org/3/reference/lexical_analysis.html#identifiers

VARIABLES

- **General**
  - Es una posición de memoria identificada por un nombre donde se almacena el valor de un dato.
  - El valor que guarda dicha variable puede cambiar a lo largo de la ejecución de un programa.
  - El tipo de variable determina:
    - Los valores que se pueden guardar en ese espacio reservado.
    - Las operaciones que se pueden realizar sobre esos valores.
    - La cantidad de memoria que se reserva para guardar ese dato.
- **Python 3**
  - No es necesario declarar las variables. Python no tiene comando para hacerlo.
  - Las variables se crean cuando se asignan por primera vez.
  - Las variables deben haberse asignado previamente antes de ser referenciadas.
  - Los valores almacenados en una variable puedes ser leídos y actualizados después de su asignación.

CONSTANTES

- **General**
  - También llamadas constantes simbólicas
  - Es un posición de memoria identificado por un nombre cuyo valor permanece inalterable a lo largo del programa y no puede modificarse.
  - Se utiliza un identificador al que se asigna un valor. Ej: PI = 3.1416 NUMMAX = 100
- **Python 3**
  - Se utilizan variables, no hay forma de declarar constantes de forma explícita.
  - Por convención se utilizan la mayúsculas para señalar aquellas variables cuyo valor no cambia en el programa.

```
# These variables are "constants" by convention:
NUMBER_OF_DAYS_IN_A_WEEK = 7
NUMBER_OF_MONTHS_IN_A_YEAR = 12
```

LITERALES

- **General**
  - También llamadas constantes literales
  - Se utilizan para expresar los diferentes valores de un dato
  - Ejemplo: “Hola”, 3.14, 45, True
- **Python 3**
  - Ejemplos de literales: numéricos, reales, imaginarios y de cadena
    - 23, 0x17, 0.2703, 0.1e-3, 7.8j, “hola”
  - https://docs.python.org/3/reference/lexical_analysis.html#literals

TIPOS DE DATOS

- **General**
  - Un dato es un símbolo físico que se utiliza para representar la información.
  - Tipos de datos:
    - Numéricos: Enteros, Reales
    - Alfanuméricos: Caracteres, Cadenas
    - Booleanos (Lógicos)
    - Listas
    - Tablas
    - Referencias, apuntadores
    - …

- **Python 3**
  - Todos los datos en un programa Python se representan por objetos o por relaciones entre objetos.
  - Tipos de datos:
    - None -> significa nada
    - Números:
      - Enteros (int) 32
      - Número en coma flotante (con parte decimal) (float) 4.7
      - Complejos (complex) 4.5 + 5.4j
    - Lógicos o Booleanos
      - En realidad son un tipo de número:
      - True y False / 0 y 1
    - Strings “hola”
    - Listas [4,5,6,7]
    - Tuplas 3,4,5
  - …
  - https://docs.python.org/3/reference/datamodel.html#objects-values-and-types
  - De los caracteres y strings se hablará en detalle más adelante

OPERADORES

- **General**
  - Los operadores permiten hacer operaciones con los datos.
  - Dependiendo del tipo de operación que realicen con los datos, los operadores se clasifican en :
    - Operadores aritméticos
    - Operadores relacionales
    - Operadores lógicos
    - Operadores de asignación
    - Operadores de Bits
    - Operadores condicionales
    - ...

- **Python 3**
  - Divide los operadores en los siguientes grupos:
    - Operadores aritméticos
    - Operadores de asignación
    - Operadores de comparación (relacionales)
    - Operadores lógicos
    - Operadores de identidad
    - Operadores de pertenecia
    - Operadores de bit
    - https://www.w3schools.com/python/python_operators.asp

OPERADORES ARITMÉTICOS

- **Python 3**

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

OPERADORES ASIGNACIÓN

- **General**
  - Consiste en asociar el valor de una expresión a una variable.
- **Python 3**

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

OPERADORES DE COMPARACIÓN (RELACIONALES)

- **Python 3**

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

OPERADORES LÓGICOS

- **General**
  - Suponiendo que A y B son expresiones lógicas las tablas de verdad son las siguientes:

A	B	A y B	A o B	no A
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

- **Python 3**

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

OPERADORES DE IDENTIDAD

- **Python 3**

  Se utilizan para comparar objetos, no si son iguales, sino si son el mismo objeto con la misma localización en memoria.

Operator	Description	Example
is	Returns true if both variables are the same object	x is y
is not	Returns true if both variables are not the same object	x is not y

OPERADORES DE PERTENENCIA

- **Python 3**
  - Se utilizan para probar si una secuencia está presente en un objeto.

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

OPERADORES DE BIT

- **Python 3**

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

EVALUACIÓN DE EXPRESIONES

- **General**
  - Una **expresión** es una combinación de operadores y operandos.
  - La **evaluación de una expresión** consiste en aplicar los operadores sobre los operandos para obtener un resultado.
  - En una expresión pueden aparecer varios operadores y unos se aplican antes que con el siguiente orden de prioridad especificado en el lenguaje.
  - Los operadores con la misma prioridad se evalúan de izquierda a derecha.

ORDE DE PRIORIDAD DE OPERADORES

- **Python 3**

Operador	Descripción
**	Exponenciación (raise to the power)
~ + -	Ccomplement, más unario y negativo (method names for the last two are +@ and -@)
* / % //	Multiplicar, dividir, y la división de módulo piso
+ -	Adición y sustracción
>> <<	Derecho y desplazamiento a la izquierda en modo bit
&	Bit a bit 'AND'td>
^ |	Bit a bit exclusiva `O 'y regulares` O'
<= < > >=	Operadores de comparación
<> == !=	operadores de igualdad
= %= /= //= -= += *= **=	Operadores de Asignación
is is not	operadores de identidad
in not in	operadores de miembros
not or and	Operadores logicos

CONVERSIONES DE TIPO

- **General**
  - Es posible transformar el tipo de una variable u objeto en otro diferente al original con el que fue declarado.
  - Este proceso se denomina “conversión” o “casting”
    - Conversiones explícitas → Las realiza el programador
    - Conversiones implícitas → Las realiza el programa de forma automática.
- **Python 3**
  - Conversiones implícitas
    - `valor=3+4.6`
  - Conversiones explícitas
    - `int('4')`
    - `float('4.2')`
    - `bool('True')`
    - `str(23)`

ENTRADA Y SALIDA DE INFORMACIÓN

- **General**
  - Cualquier programa informático debe contar con una entrada de datos y una salida de información.
  - La información puede a y desde la consola, pero también puede tomarse de ficheros o bases de datos.
- **Python 3**
  - Para la entrada de información desde consola utilizaremos la función input.
  - Input tiene como parámetro opcional el texto que se escribirá en la consola para indicar al usuario que introduzca los datos.
  - Cuando se llama a la función, el programa se para hasta que el usuario introduce un valor y ha pulsado “intro” → En ese momento el programa continúa su ejecución.

ENTRADA DE INFORMACIÓN PYTHON3

- **Python 3**
  - La función input devuelve la entrada del usuario en un string sin ningún cambio.
  - Si queremos que la cadena obtenida sea transformada a otro tipo de datos, debemos hacer una conversión de datos con un cast o la función eval.
  - Ejemplo:

```
>>> population = int(input("Population of Toronto? "))
Population of Toronto? 2615069
>>> print(population, type(population))
2615069 <class 'int'>
>>>
```

SALIDA DE INFORMACIÓN

- **Python 3**
  - Para la salida de datos a consola utilizamos la palabra print.
  - Para combinar texto y variables usamos coma (,)
  - Si son cadenas se puede usar el +
  -

```
x = "awesome"
print("Python is " + x)
```

```
x = "Python is "
y = "awesome"
z =  x + y
print(z)
```

https://docs.python.org/3/library/functions.html#print

ESTRUCTURAS DE CONTROL DE FLUJO

- También llamado “Teorema Fundamental de la Programación Estructurada”.
- El teorema de Böhm - Jacopini [C.Böhm, G.Jacopini, Comm. ACM vol.9, nº5, 366-371, 1966] establece que:
- **“Todo programa propio se puede escribir utilizando únicamente las estructuras de control secuencial, condicional e iterativa“.**
- Un programa se define como propio si cumple las siguientes condiciones:
  - Si tiene un solo punto de entrada y un solo punto de salida
  - Existen caminos desde la entrada hasta la salida que pasan por todas las partes del programa.
  - Todas las instrucciones son ejecutables y no existen bucles sin fin.

ESTRUCTURA DE CONTROL DE SECUENCIAL

- **General**
  - Una estructura secuencial está formada por un conjunto de sentencias (instrucciones) que se ejecutan una detrás de otra.
    - Ejemplo
      - Sentencia 1
      - Sentencia 2
      - …
      - Sentencia N
- Python 3

```
#Primer programa en python
#Leer los datos
n1 = int(input('Primer numero: '))
n2 = int(input('Segundo numero: '))
#Realizar los calculos
suma=n1+n2
#Obtener el resultado
print('La suma de los numeros es')
print suma
```

ESTRUCTURA DE CONTROL CONDICIONAL

- **General**
  - También de selección, alternativa, de bifurcación
  - Se utiliza cuando en un determinado punto de un programa se requiere que se elija entre dos o más acciones distintas, en función de alguna condición determinada.

```
SI <condición> ENTONCES
    <Acción 1>
FIN SI
```

```
SI <condición> ENTONCES
    <Acción 1>
EN OTRO CASO
    <Acción 2>
FIN SI
```

ESTRUCTURA DE CONTROL CONDICIONAL

- **Python 3**
  - IMPORTANTE: Python necesita la indentación, usando espacios en blanco, para definir el ámbito en el código

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

ESTRUCTURA DE CONTROL REPETITIVA

- GENERAL
  - También cíclica, iterativa o de bucle
  - Se repite un número determinado de veces mientras se cumple una determinada condición
  - Cada vez que se ejecuta el cuerpo del bucle, se dice que se ha producido una iteración
  - La salida del bucle se produce cuando se ha dejado de cumplir la condición de repetición

```
MIENTRAS <Condición> HACER
    <Acción 1>
FINMIENTRAS
```

ESTRUCTURA DE CONTROL REPETITIVA

- IMPORTANTE: Python necesita la indentación, usando espacios en blanco, para definir el ámbito en el código

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

MÁS EN PRÓXIMAS PRESENTACIONES