Python_1

February 25, 2020

1 Introducción a Programación

Bienvenido. En este curso vamos aprender programación usando el lenguaje python. Vamos a empezar con el clasico programa de introducción. El "hola mundo". Este es un programa que al ejecutarse muestra el texto "hola mundo" por pantalla. Para ejecutar una celda de codigo pulsa shift + enter. El programa es el siguiente:

[ ]:
```
print("hola mundo")
```

El codigo anterior nos introduce a la *funcion* **print**. Está es la orden de python para mostrar cosas por pantalla, ya sean texto, numeros o una mezcla de ellos. `print("hola mundo")` muestra el texto *hola mundo* por pantalla. En programación para diferenciar lo que es un texto de los que es parte del programa se usa el simbolo dobles comillas " ". Todo lo que este dentro de" " serà entendido como texto, no como código.

Por ejemplo el mismo código anterior pero sin el correcto uso de las dobles comillas nos da el siguiente error.

[ ]:
```
print(hola mundo)
```

Error de sintaxi en la linea 1 de código. En este caso solo tenemos una asi que solo puede ser en la primera.

1.0.1 Tu turno

Escribe un programa en la siguiente celda que muestre por pantalla tu nombre

[1]:
```
print("Joan Marcual")
```

```
Joan Marcual
```

Escribe un programa el el que se muestren dos lineas, una con tu nombre y la siguiente con tus apellidos

[4]:
```
#Hay dos opciones

#Dos prints
print("Joan")
print("Marcual")
```

```
#Un print con el caracter especial \n(salto de linea)
print("Joan\nMarcual")
```

```
Joan
Marcual
Joan
Marcual
```

1.1 Variables

En todo lenguaje de programación se pueden declarar variables, por ejemplo `x = 42`. De esta forma podemos guardar, modificar y operar con datos. Por ejemplo el siguiente código crea 3 variables **x, y, z** con los valores **1, 2 , 3**

Nota: python no es un lenguaje tipado (untyped). Lo que significa que python no te hace definir un tipo especifico de variable, por contra en un lengauge tipado clasicamente deberiamos declar x como `int x` antes de asignarle un numero entero.

[ ]:
```
x = 1
y = 2
z = 3
```

Ahora que tenemos estas varibles, podemos trabajar con ellas por ejemplo podemos mostrarlas. Haciendo uso de las función `print`

[ ]:
```
print(x)
print(y)
print(z)
```

Las variables pueden ser de varios tipos distintos, algunos de ellos son: * **Números**: como `42` * **Boleanos**: como `True` o como `False` * **Texto**: como `"hola mundo"`

1.1.1 Tu turno

En la siguiente celda, declara una variable llamada `miTexto` con el contenido "hola mundo". Y muéstrala por pantalla

[5]:
```
miTexto = "hola mundo"
print(miTexto)
```

```
hola mundo
```

Crea una varibale booleana que crea sea cierta, `True` y muéstrala por pantalla

[6]:
```
cierto = True
```

1.2 Operaciónes con variables

Una vez tenemos variables definindas podemos operar con ellas. Por ejemplo podemos hacer operaciones matematicas con las variables numericas. El la siguiente celda declaramos dos vari-

ables, las sumamos guardandonos el resultado en una tercera variable y mostramos esta última por pantalla.

[ ]:
```
x = 3
y = 4

z = x + y
print(z)
```

1.2.1 Tu turno

Crea un código que cree 3 variables, **x y z** con los valores **4 5 2**. Sume las x e y. Divida por z, y muestre el resultado por pantalla.

[9]:
```
x = 4
y = 5
z = 2
resultado = (x+y)/z #ojo con las prioridades de operaciones,
                    #sin el parentesis se ejecutara la division primero
print(resultado)
```

```
4.5
```

Declara dos variables de texto. Por ejemplo una con el contenido "hola" y la otra con el contenido "adios", sumalas y muéstralas por pantalla.

[12]:
```
texto1 = "hola"
texto2 = "adios"
sumaDeTextos = texto1 + texto2
print(sumaDeTextos)
#Si queremos un espacio entre medio de las dos palabras, podemos sumar el texto␣
 ↪espacio
sumaDeTextos = texto1 + " " + texto2
print(sumaDeTextos)
```

```
holaadios
hola adios
```

Ahora prueba de sumar una variable de texto con una variable numérica

[14]:
```
t = "hola"
x = 7
x + t
# Las operaciones entre diferentes tipos no funcionan normalmente
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)
```

```
<ipython-input-14-9aa25629d20a> in <module>
  1 t = "hola"
  2 x = 7
----> 3 x + t
  4 # Las operaciones entre diferentes tipos no funcionan normalmente


TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

1.2.2 Impresión de variables junto a texto

Si queremos mostrar por pantalla una variable junto a un texto, usaremos la función `format` al final del texto usando un punto. En el texto señalizaremos dónde queremos que `format` nos ponga las variables usando los simbolos {}. Y por último le diremos a `format` que variables poner.

[ ]:
```
a = 99
print("El valor de la variable a es {}".format(a))
a = 11
print("Encambio el valor de a ahora es {}".format(a))
```

Podemos mostrar tantas variables como queramos a la vez. Por ejemplo

[ ]:
```
a = 1
b = 2
print("a es {} y b es {}".format(a,b))
```

1.2.3 Tu turno

Define dos variables, una con tu nombre y otra con tu edad. Usando la función `format` haz que muestre por pantalla la frase "*tuNombre* tiene *edad* años"

[15]:
```
tuNombre = "pepe"
edad     = 40
print("{} tiene {} años".format(tuNombre, edad))
```

```
pepe tiene 40 años
```

1.3 Operaciones matemáticas simples

Las operaciones matemáticas más comunes son la suma, la resta, la múltiplicación y la división. Estas operación se señalizan con los clasicos simbolos `+ - * /` A estas 4 operaciones debemos añadir la operacion *módulo* y la operación *division entera*. La division entera es la división sin parte decimal se denota con el simbolo `//`, y el *modulo* hace es cálcular el residuo de una división. La operación módulo se denota con el símbolo `%`.

Nota: Cuando una línea de código es demasido larga la podemos cortar haciento uso del simbolo `\`

[ ]:
```
dividendo = 10
divisor   = 4
quociente = dividendo//divisor
residuo   = dividendo%divisor

print("El residuo de la division de {} y {} es {}, y el residuo \
es {}".format(dividendo, divisor, quociente, residuo))
```

1.3.1 Tu turno

Crea una variable x con valor 1595, y una variable y con valor 1660. Calcula usando la operación módulo si estos numeros son divisibles por 11

[18]:
```
x = 1595
y = 1660
residuoX = x % 11
residuoY = y % 11
print("El resiudo de dividir {} entre 11 es {}".format(x,residuoX))
print("El resiudo de dividir {} entre 11 es {}".format(y,residuoY))
```

```
El resiudo de dividir 1595 entre 11 es 0
El resiudo de dividir 1660 entre 11 es 10
```

1.4 Entrada

La programación no sirve de nada si no podemos entrar los datos que nos interesan calcular. Para ello existen las operaciones de entrada. Ahora veremos como leer datos que entren desde el teclado y guardarlos en una variable.

[ ]:
```
print('Escribe tu nombre:')
x = input()
print("Hola {}, bienvenido.".format(x))
```

1.4.1 Tu turno

Crea un programa que pregunte tu nombre, despues tu apellido, y te de la bienvenida con tu nombre completo

[19]:
```
print("Escribe tu nombre")
nombre = input()
print("Escribe tu apellido")
apellido = input()
print("Hola {} {}, bienvenido.".format(nombre, apellido))
```

```
Escribe tu nombre
joan
Escribe tu apellido
```

```
marcual
Hola joan marcual, bienvenido.
```

Crea un programa donde puedas entrar numeros y te diga si estos son pares o no. *Nota: por defecto python se piensa que lo que los valores que le entran son texto, para interpretar las entradas como numeros debemos decir que son numeros los valores que entran. Lo podemos hacer de la siguiente forma* `int(input())`

[20]:
```
x = int(input())
print(x%2==0)
```

```
5
False
```

Crea un programa donde entren dos numeros y te devuelva su multiplicación

1.5 Operaciones booleanenas

Hemos visto que las variables tambien pueden tomar valores booleanos, estos son `True` y `False`. Con ellos podemos usar operaciones de lógica como por ejemplo `and`, `or`. Recordemos cuales son las tablas de la verdad de estas operaciones. El resultado de la and es solo cierto cuando las dos variables son ciertas y falso en todos los demás casos. Y el resultado de la or es cierto cuando almenos una de las dos es cierta.

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

Con ellas podemos modelar eventos. En el siguiente ejemplo modelamos si alguien saldra o no en a montar en bici. Para ello usamos el operador logico `and`

[1]:
```
haceSol     = False
tiempoLibre = True

salirEnBici = haceSol and tiempoLibre
print(salirEnBici)
```

```
False
```

1.5.1 Tu turno

Usando operadores lógicos rellena el siguiente código para decidir si ir o no a una fiesta. Queremos ir a la fiesta si va uno de nuestros amigos o si van dos conocido. Rellena la definición de la

variable *irFiesta* para que actue como deseamos.

[8]:
```
amigo = False
conocido1 = False
conocido2 = False
#Esta es la linea que tienes que rellenar
irFiesta = amigo or (conocido1 and conocido2)
print("Ire a la fiesta es igual a {}".format(irFiesta))
```

```
Ire a la fiesta es igual a False
```

1.6 Comentarios

Los comentarios sirven para añadir líneas en el código que no serán ejecutadas. Sirven mayoritariamente para explicar lo que se está haciendo y así hacer el código más fácil de compreender. Esto se hace con el símbolo #, después de # nada será ejecutado en esa línea. Si queremos usar esta operación en bloque usaremos tres veces dobles comillas """ y cerraremos de la misma forma """

[ ]:
```
a = 2 #Aqui estamos asignando 2 a la variable a
# a = 3 esto no se ejecuta
"""
Si queremos
varias
lineas
de comentarios
lo podemos hacer asi
"""
print(a)
```

1.7 Funciones

Cuando un trozo de código será, o tiene potencial de ser, usado más de una vez lo empaquetaremos en una *función*. Las funciones sirven para no tener que reescribir el mismo código una y otra vez cada vez que queremos implementar esa funcionalidad. También nos sirven para separar las tareas de programación entre diferentes programadores y despreocuparnos de como se hacen ciertas operaciones.

Para crear o definir una función usaremos la palabra clave `def`. Junto al nombre que le demos a la función, y entre parentesis `()` las variables que usará seguido de dos puntos `:`. Dejaremos los parentesis vácios en caso de que no usemos ninguna.

Ejemplo de función.

[11]:
```
def darLaBienvenida(nombre):
    print("Bienvenido {}".format(nombre))

nombre1 = "Fulanito"
nombre2 = "Pepito"
darLaBienvenida(nombre1)
```

```
darLaBienvenida(nombre2)
```

```
Bienvenido Fulanito
Bienvenido Pepito
```

Ejemplo 2. En este caso hacemos que la funcion *retorne* un valor. Este valor podrá ser asignado a una variable como en el siguiente ejemplo.

[13]:
```
def calcularArea(base, altura):
    area = base * altura
    return area

print("Introduce la base")
miBase = int(input()) #Se debe usar int(input()) para leer numeros enteros
print("Introduce la altura")
miAltura = int(input())

miArea = calcularArea(miBase, miAltura)
print(miArea)
```

```
Introduce la base
3
Introduce la altura
4
12
```

Fijate que el código dentro de la función deberá estar identado, esto significa que deberá estar 4 espacios a la derecha, o una tabulación (tecla Tab).

1.7.1 Tu turno

Crea la función *suma* que dados dos numero devuelva su suma

[21]:
```
def suma(a,b):
    resultado = a + b
    return resultado

print(suma(10,5))
```

```
15
```

Crea la función *media* que dados 4 numeros devuelva su media

[23]:
```
def media(a, b, c, d):
    sumaABCD = a + b + c + d
    resultado = sumaABCD / 4
    return resultado

print(media(1,2,3,4))
```

```
10.0
```

Crea una función areaTriangulo que calculo areas de triangulos dadas la base y la altura

[25]:
```
def calcularAreaTriangulo(base, altura):
    area = (base * altura)/2
    return area

print(calcularAreaTriangulo(2,5))
```

```
5.0
```

1.8 Control de flujo I

Muchas veces nos encontramos que dependiendo de los valores que tenemos queremos que los programas ejecuten cálculos diferentes. Esto lo podemos hacer con la palabra clave `if`, *si* condicional en español.

En el ejemplo siguiente creamos la función aprovar. Que mirará si la nota es mayor que o igual que cinco y nos dirá por pantalla si hemos aprobado o suspendido

[27]:
```
def aprobar(nota):
    if nota >= 5:
        print("Has aprobado!")
    if nota < 5:
        print("Lo siento has suspendido")

print("Introduce tu nota:")
miNota = int(input());
aprobar(miNota)
```

```
Introduce tu nota:
4
Lo siento has suspendido
```

1.8.1 Tu turno

Crea una función evaluar, que escriba por pantalla. Suspendio si la nota es menor a 5, suficiente si es de 5 a 6, bien si es de 6 a 7, notable si es de 7 a 9, y excelente si es mayor de 9.

[ ]:
```
def notaATexto(nota):
    if nota < 5:
        print("Suspendido")
    if nota >= 5 and nota < 6:
        print("Suficiente")
    if 6>=nota<7:
        print("Bien")
    if nota >=7 and nota <9:
        print("Notable")
```

```
    if nota >= 9:
        print("Excelente")

notaATexto(int(input()))
```

1.9 Control de flujo II

Además de la palabra clave `if` también existe la palabra clave `else` que se ejecuta en caso de que la condición del if no se cumpla. En español se podria leer como "por contra ejecuta esto".

El ejemplo anterior puede ser reescrito de esta forma usando `else`

[ ]:
```
def aprobar(nota):
    if nota >= 5:
        print("Has aprobado!")
    else:
        print("Lo siento has suspendido")

print("Introduce tu nota:")
miNota = int(input());
aprobar(miNota)
```

```
Introduce tu nota:
```

También existe la palabra clave `elif` que sirve para comprobar una condición en caso de que el anterior if no sea cierto. Veamos esto en los siguientes ejemplos.

Por ejemplo queremos hacer una función que nos transforme la nota de numérica a escala de "bien, notable, etc"

[1]:
```
def imprimirNotaTexto(nota):
    if nota >= 9:
        print("excelente")
    elif nota >= 7:
        print("notable")
    elif nota >= 6:
        print("bien")
    elif nota >= 5:
        print("suficiente")
    elif nota < 5:
        print ("suspenso")


print("Introduce la nota")
nota = int(input())
imprimirNotaTexto(nota)
```

```
Introduce la nota
9
```

```
excelente
```

1.9.1 Tu turno

Ahora vamos a cambiar los `elif` por `if`. Prueba con las siguientes notas: **3,5,6,7,10** y explicad que ocurre.

[2]:
```
def ERRONEA_transformarNota2Texto(nota):
    if nota >= 9:
        print("excelente")
    if nota >= 7:
        print("notable")
    if nota >= 6:
        print("bien")
    if nota >= 5:
        print("suficiente")
    if nota < 5:
        print("suspenso")

print("Introduce la nota")
nota = int(input())
ERRONEA_transformarNota2Texto(nota)
```

```
Introduce la nota
9
excelente
notable
bien
suficiente
```

Escribe tu explicación a continuación:

Crea una función usando `elif` que transforme le entren los grados centigrados y imprima por pantalla si hace **Mucho calor, calor, buena temperatura, fresquibiris, frio, o frio que pela**, dependiendo de los grados. Decide tu mismo las temperaturas para cada texto.

[6]:
```
def temperatura(grados):
    if(grados > 40):
        print("Mucho calor")
    elif(grados > 30):
        print("calor")
    elif(grados > 20):
        print("buena temperatura")
    elif(grados > 15):
        print("fresquibiris")
    elif(grados > 0):
        print("frio")
    else:
        print("frio que pela")
```

```
grados= int(input("Cuantos grados hace?"))
temperatura(grados)
```

```
Cuantos grados hace?-50
frio que pela
```

Crea una función que le entren dos parametros. Uno será una nota numérica ej:5,6,7 y el otro será una temperatura en grados por ejemplo:15,17,23. Crea una funcion que haga lo siguiente. Si la nota es menor que 5 saque por pantalla el texto *Toca estudiar*. Si la nota es mayor que cinco y la temperatura es mayor que 15 grados saque por pantalla *ir de excursion*. Si la nota es mayor que 5 y la temperatura es menor que 0 grados imprima *quedarse en casa*. En cualquier otro caso que imprima *ir al cine*.

Este ejercicio se puede implementar de dos formars distintas usando operaciones booleanas (and or not) o anidando condificones `if`. Esto significa poniendo if dentro de otros `if`s. Crea ambas soluciones.

[9]:
```
def queHacer(nota, grados):
    if nota < 5:
        print("toca estudiar")
    else:
        if grados > 15:
            print("ir de excursion")
        elif grados < 0:
            print("quedarse en casa")
        else:
            print("ir al cine")
miNota= int(input())
grados = int(input())
queHacer(grados=grados, nota=miNota)
```

```
4
20
toca estudiar
```

1.10 Control de flujo II

1.10.1 Bucles

Es muy en programación querer realizar una misma operación varias veces. Para ello se usan lo que se denominan bucles. Un bucle es una sección de código que se repetirá hasta alcanzar la condición definida. Para ello podemos hacer use de las intrucciones. `while` y `for`. Ambas instucciones sirven para lo mismo, pero simplemente se expresan de forma distinta. Para elegir cual usar en cada ocasión simplemente usaremos la que nos resulte más natural para el problema.

Ejemplo while

Sacamos por pantalla el valor de i mientras i sea mas pequeña que 6

[10]:
```
i = 1
while i < 6:
    print(i)
    i += 1
```

```
1
2
3
4
5
```

En este caso usamos la instrucción `range`, rango en español, donde definimos desde y hasta donde queremos que se ejecute el bucle. En este caso la instrucción `for` se encarga de definir la variable **i** y la instrucción `range` se encarga de incrementar en cada iteración del bucle la variable **i**.

[12]:
```
for i in range(10,15):
    print(i)
```

```
10
11
12
13
14
```

1.10.2 Tu turno

Crea una función que sume todos los números del 1 al 100. Y muestre por pantalla la suma.

[18]:
```
def sumaSucesion(n):
    suma = 0
    for i in range(0,n+1):
        suma = suma + i
    return suma
print(sumaSucesion(100))
```

```
5050
```

Crea una función que muestre por pantalla todos los números múltiples de 5 entre los números 0 y 100.

[23]:
```
def multiples5(n):
    for i in range(1,n+1):
        if(i%5==0):
            print(i)
multiples5(100)
```

```
5
10
15
```

```
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
```

Crea una función que pida números mientras el usuario escriba número mayores que el primero.

[2]:
```
def numMayor():
    texto = "introduce un numero: "
    primero = int(input(texto))
    actual  = int(input(texto))
    while(actual > primero):
        actual  = int(input(texto))
    print("FIN")

numMayor()
```

```
introduce un numero: 2
introduce un numero: 3
introduce un numero: 4
introduce un numero: 5
introduce un numero: 4
introduce un numero: 1
FIN
```

Crea una función que pida números mientras el usuario escriba número mayores que el anterior.

[30]:
```
def numMayorAnterior():
    texto    = "introduce un numero: "
    anterior = int(input(texto))
    actual   = int(input(texto))
    while(actual > anterior):
        #print("anterior vale {}".format(anterior))
        #print("actual vale {}".format(actual))
        anterior = actual
        actual   = int(input(texto))
```

```
    print("FIN")

numMayorAnterior()
```

```
introduce un numero: 5
introduce un numero: 6
introduce un numero: 5
FIN
```

Crea una función que pida números mientras no se escriba un número negativo. El programa terminará escribiendo la suma de los números introducidos.

[9]:
```
def numHastaNegativo():
    texto  = "introduce un numero: "
    actual = int(input(texto))
    suma   = 0
    while(actual >= 0):
        suma = suma + actual
        actual = int(input(texto))

    print("FIN")
    print("La suma de los numeros introuducidos es {}".format(suma))
numHastaNegativo()
```

```
introduce un numero: 5
introduce un numero: 6
introduce un numero: -8
FIN
La suma de los numeros introuducidos es 11
```

Crea una función que pida números mientras no se escriba un número negativo. La función al acabar tiene que devolver la cantidad de numeros multiples de 3 que han sido introducidos.

[32]:
```
def multiples3():
    texto    = "introduce un numero: "
    actual   = int(input(texto))
    contadorMultiplesDe3 = 0
    while(actual >= 0):
        if(actual%3==0):
            contadorMultiplesDe3 = contadorMultiplesDe3 + 1
        actual   = int(input(texto))
    print("FIN")
    print("Has introducido {} multiplos de 3".format(contadorMultiplesDe3))
multiples3()
```

```
introduce un numero: 2
introduce un numero: 3
introduce un numero: 4
introduce un numero: 5
```

```
introduce un numero: -1
FIN
Has introuducido 1 multiplos de 3
```

Crea una función que entrado un número diga si este número es primo

[12]:
```python
def esPrimo(n):
    for i in range(2,n):
        if(n%i==0):
            return False
    return True

numero = int(input("introduce un numero: "))
print(esPrimo(numero))
```

```
introduce un numero: 113
True
```

Crea una función que entrando un número imprima todos sus divisores (número por los que puede ser divido)

Crea una función que dado un texto y una letra, diga en que posición se encuentra la primera aparición de esta letra

[22]:
```python
#Devuelve -1 en caso de que no exista
def primeraLetra(texto, letra):
    indice = 0
    for l in texto:
        if l == letra:
            return indice
        indice = indice + 1
    return -1

texto = "supercalifragilisticoespialidoso"
letra = "a"
print(primeraLetra(texto,letra))
```

```
6
```

Crea una función que cuente el número de apariciones de una letra en un texto

[23]:
```python
def contarLetra(texto, letra):
    contador = 0
    for l in texto:
        if l == letra:
            contador = contador + 1
    return contador

texto = "supercalifragilisticoespialidoso"
letra = "a"
```

```
print(contarLetra(texto,letra))
```

```
3
```

Crea una función que imprima tantos numero de la sequencia de fibonacci como desees.

[25]:
```
def fibonacci(n):
    anterior = 1
    actual = 1
    for i in range(0,n):
        print(actual)
        aux = actual
        actual = anterior + actual
        anterior = aux
fibonacci(11)
```

```
1
2
3
5
8
13
21
34
55
89
144
```

[ ]:
```

```