# FUNCIONES

I. Gracia, P. García, A. López
Junio, 2023

# Lo que ya sabíamos

**aceite.py**

```
from math import ceil

# Entrada de datos
litros = float(input('¿Cuántos litros necesitas? '))

# Cálculos
botellas = ceil(litros / 0.75)
precio_botellas = botellas * 5.25
garrafas = ceil(litros / 5)
precio_garrafas = garrafas * 27.50

# Mostrar resultados
print(f'{botellas} botellas cuestan {precio_botellas} euros')
print(f'{garrafas} garrafas cuestan {precio_garrafas} euros')
```

Sabemos hacer uso de funciones predefinidas

# Lo que ya sabíamos

**aceite.py**

```
from math import ceil

# Entrada de datos
litros = float(input('¿Cuántos litros necesitas? '))

# Cálculos
botellas = ceil(litros / 0.75)
precio_botellas = botellas * 5.25
garrafas = ceil(litros / 5)
precio_garrafas = garrafas * 27.50

# Mostrar resultados
print(f'{botellas} botellas cuestan {precio_botellas} euros')
print(f'{garrafas} garrafas cuestan {precio_garrafas} euros')
```

y también de funciones definidas en módulos

# Funciones definidas en un programa

**imc_con_función.py**

```
# Definición de funciones
def calcular_imc(altura, peso):
    return peso / (altura ** 2)

# Programa principal
metros = float(input('Introduce la altura (en metros): '))
kilos = float(input('Introduce el peso (en kilos): '))

imc = calcular_imc(metros, kilos)

print(f'El IMC es {imc:0.2f}')
```

Definición de la función

Llamada a la función

# Programa principal

**imc_con_función.py**

```
# Definición de funciones

def calcular_imc(altura, peso):
    return peso / (altura ** 2)

# Programa principal
metros = float(input('Introduce la altura (en metros): '))
kilos = float(input('Introduce el peso (en kilos): '))

imc =  calcular_imc(metros, kilos)

print(f'El IMC es {imc:0.2f}')
```

Programa principal

# Programa con varias funciones

**nivel_peso_con_funciones.py**

```
# Definición de funciones
def calcular_imc(altura, peso):
    return peso / (altura ** 2)

def calcular_nivel_peso(altura, peso):
    # Cálculos para hallar el nivel de peso
    return nivel

# Programa principal
# (No se muestran sus instrucciones)
```

Nos centraremos en esta función

# Definición de una función

**nivel_peso_con_funciones.py**

```
def calcular_nivel_peso(altura, peso):
    imc = calcular_imc(altura, peso)
    if imc < 18.5:
        nivel = 'bajo'
    elif imc < 25:
        nivel = 'normal'
    elif imc < 30:
        nivel = 'sobrepeso'
    else:
        nivel = 'obesidad'
    return nivel
```

Tiene este aspecto:

```
def nombre_función(parámetros):
    instrucciones
    return valor
```

Cabecera

Cuerpo

# Cabecera de una función

**nivel_peso_con_funciones.py**

```
def calcular_nivel_peso(altura, peso):
    imc = calcular_imc(altura, peso)
    if imc < 18.5:
        nivel = 'bajo'
    elif imc < 25:
        nivel = 'normal'
    elif imc < 30:
        nivel = 'sobrepeso'
    else:
        nivel = 'obesidad'
    return nivel
```

Tiene este aspecto:

```
def nombre_función(parámetros):
    instrucciones
    return valor
```

Datos de entrada de la función

# Valor de retorno

**nivel_peso_con_funciones.py**

```
def calcular_nivel_peso(altura, peso):
    imc = calcular_imc(altura, peso)
    if imc < 18.5:
        nivel = 'bajo'
    elif imc < 25:
        nivel = 'normal'
    elif imc < 30:
        nivel = 'sobrepeso'
    else:
        nivel = 'obesidad'
    return nivel
```

Tiene este aspecto:

```
def nombre_función(parámetros):
    instrucciones
    return valor
```

Dato de salida de la función

# Cuerpo de una función

**nivel_peso_con_funciones.py**

```
def calcular_nivel_peso(altura, peso):
    imc = calcular_imc(altura, peso)

    if imc < 18.5:
        nivel = 'bajo'
    elif imc < 25:
        nivel = 'normal'
    elif imc < 30:
        nivel = 'sobrepeso'
    else:
        nivel = 'obesidad'

    return nivel
```

Tiene este aspecto:

```
def nombre_función(parámetros):
    instrucciones
    return valor
```

El cuerpo contiene una llamada a otra función y una sentencia condicional múltiple

# Cuerpo de una función

**factorial.py**

```
def factorial(n):
    producto = 1
    i = 1
    while i <= n:
        producto = producto * i
        i = i + 1
    return producto

# Programa principal
# (No se muestran sus instrucciones)
```

Tiene este aspecto:

```
def nombre_función(parámetros):
    instrucciones
    return valor
```

Este otro cuerpo contiene un bucle

# Parámetros y argumentos

**imc_con_función.py**

```
# Definición de funciones

def calcular_imc( altura , peso ):
    return peso / (altura ** 2)

# Programa principal
metros = float(input('Introduce la altura (en metros): '))
kilos = float(input('Introduce el peso (en kilos): '))

imc =  calcular_imc( metros , kilos )

print(f'El IMC es {imc:0.2f}')
```

Parámetros

Argumentos

# Parámetros y argumentos

**imc_con_función.py**

```
# Definición de funciones

def calcular_imc( altura , 
    return peso / (altura 

# Programa principal
metros = float(input('Introdu    la altura (en metros): '))
kilos = float(input('Intro uce el peso (en kilos): '))

imc = calcular_imc(metros,kilos)

print(f'El IMC es {imc:0.2f}')
```

Podemos hacer llamadas así

```
calcular_imc(1.20, 21.7)
```

# Paso de parámetros

**imc_con_función.py**

```
# Definición de funciones

def calcular_imc( altura , peso ):
    return peso / (altura ** 2)

# Programa principal
metros = float(input('Introduce la altura (en metros): '))
kilos = float(input('Introduce el peso (en kilos): '))

imc =  calcular_imc( metros , kilos )

print(f'El IMC es {imc:0.2f}')
```

Cada argumento se asigna al parámetro correspondiente

# Ejemplo: número combinatorio

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Necesitamos calcular tres factoriales

# Ejemplo: número combinatorio

**combinatorio.py**

```
# Definición de funciones
def factorial(n):
    # No se muestra de nuevo el cuerpo de esta función

def combinatorio(n, k):
    return factorial(n) // (factorial(k) * factorial(n - k))

# Programa principal
n = int(input('Introduce el número n: '))
k = int(input('Introduce el número k: '))
print(f'El número {n} sobre {k} es {combinatorio(n, k)}')
```

# Ejemplo: número combinatorio (cont.)

**combinatorio_engorroso.py**

```
def combinatorio(n, k):
    # Calcular factorial de n
    fact_n = 1
    i = 1
    while i <= n:
        fact_n = fact_n * i
        i = i + 1
    # Calcular factorial de k
    fact_k = 1
    i = 1
    while i <= k:
        fact_k = fact_k * i
        i = i + 1
    # Debajo del código de la izquierda
    # Calcular factorial de n - k
    fact_n_k = 1
    i = 1
    while i <= n - k:
        fact_n_k = fact_n_k * i
        i = i + 1
    # Devolver el número combinatorio
    return fact_n // (fact_k * fact_n_k)

# Programa principal
# (No se muestran sus instrucciones)
```

Isabel Gracia, Pedro García-Sevilla, Ángeles López
gracia@uji.es, pgarcia@uji.es, lopeza@uji.es