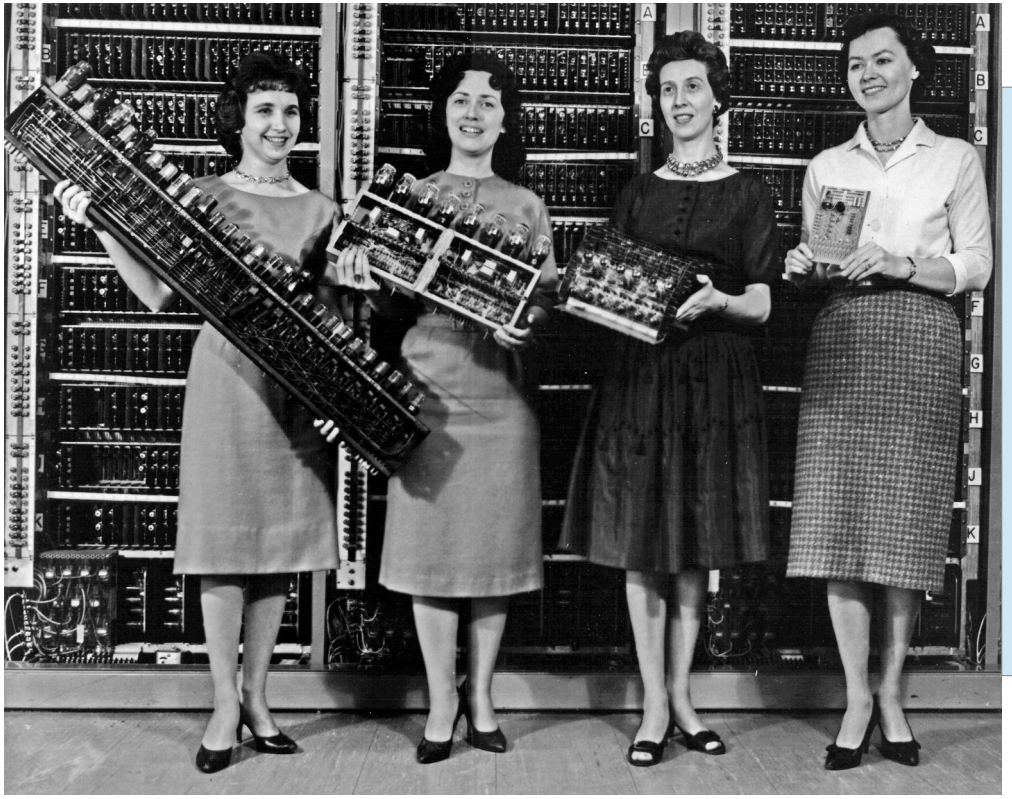
FUNCIÓNS

EN

PYTHON

Programadoras do ENIAC



- **Betty Snyder Holberton (1917-2001)**
- **Betty J. Jennings Bartik (1924-2011)**
- **Ruth L. Teitelbaum (1924-1986)**
- **Kathleen McNulty (1921-2006)**
- **Frances Bilas Spence (1922-2012)**
- **Marlyn Wescoff Meltzer (1922-2008).**

- ENIAC: Electronic Numerical Integrator And Computer
- Primeiro ordenador de propósito xeral (1946-1955)
- Elas desenvolveron as bases da programación de ordenadores

Definición de funcións

- Funcións é un recurso fundamental na programación que permite agrupar expresións e sentencias, que só se executa cando a función é invocada ou chamada.
- O uso de funcións ten asociado dúas accións:
  - **Definición da función**: contén o conxunto de sentencias que especifica a tarefa que realiza a función.
  - **Chamada da función**: executa a tarefa especificada pola función sobre uns datos dados.

¿Cando utilizar funcións?

- Recomendacións:
  - Encapsular nunha función bloques de código que se repitan en varias partes do programa (aínda que sexa operando sobre datos distintos). Evita copiar e pegar código.
  - Definir como función bloques de código que realicen unha tarefa definida que poidamos nomear. Fai o código máis lexible.
  - Cando temos unha función que fai demasiadas tarefas, dividir función en funcións máis pequenas que fagan unha tarefa simple utilizando estratexias de deseño descendente.

Definición de funcións propias

- Sintase:

```
# definicion de funcion propia
# argEntrada1, argEntrada2, ... son os argumentos de entrada
def nome_funcion(argEntrada1, argEntrada2, ...):
    sentencias co corpo da funcion
    # despois de return vai a lista de argumentos
    # que devolve a funcion
    return [argSaida1, argSaida2, ...]

# chamada ou invocacion a funcion
[a1, a2, ...] = nome_funcion(arg1, arg2,...)
```

- Na primeira liña: palabra reservada **`def`**, un nome de función (coas regras dos nomes de variables), os argumentos de entrada entre parénteses e remata con :
- Despois ven o corpo da función con tabulación.
- Remata coa palabra reservada **`return`** seguida por unha lista cos argumentos de saída.

Definición de funcións propias

- A definición da función remata coa tabulación (de igual xeito que nas sentencias de selección e iteracción).
- Coa definición da función non se executa o corpo da función. Para executar o corpo da función hai que facer unha chamada a función.
- A chamada a función pódese facer desde o programa principal, desde o interprete de Python ou desde outra función.
- As variables definidas no corpo da función son locais a función e non se pode acceder a elas desde código que non estea no corpo da función.
- Unha función realiza unha tarefa concreta, e antes da súa definición, hai que identificar os argumentos de entrada (datos que necesita a función para levar a cabo a tarefa) e os argumentos de saída (resultados que se espera da execución da tarefa).
- Os argumentos de entrada teñen que ter un valor antes de invocar a función.

Variables globais

- Desde unha función pódense ler as variables do programa principal (neste caso x e y) que invocou a función:

```
x=5; y=2
# funcion que multiplica dous números
def multiplica():
    z=x*y
    return z
# chamada de funcion
print('x*y= ', multiplica())
print('x= ', x, 'y= ',y)
```

- Se modificas unha variable do programa principal, modificas unha variable que se chama igual pero local da función

```
x=5; y=2
# funcion que multiplica dous números
def multiplica():
    z=x*y
    x=6; y=3
    return z
# chamada de funcion
print('x*y= ', multiplica())
print('x= ', x, 'y= ',y)
```

Variable y local da función multiplica( ), non é a variable y=2 do programa principal

Variables globais

- Esto ocurre porque as variables son locais ó seu ámbito de definición e non son compartidas entre o programa principal e o corpo da función.
- Os argumentos de entrada dunha función tamén son variables locais a función.
- Para aceder a unha variable do programa principal desde unha función hai que declarala como global coa palabra reservada **`global`** (dentro do corpo da función):

```
x=5; y=2
# funcion que multiplica dous números
def multiplica():
    # declaracion de x e y como variables non locais a funcion
    global x, y
    z=x*y
    x=6; y=3
    return z
# chamada de funcion
print('x*y= ', multiplica())
print('x= ', x, 'y= ',y)
```

Funcións lambda

- Son funcións que se poden definir nunha soa liña e chámanse funcións **`lambda`**. Exemplo:

```
# definicion de funcion
def f(x):
    return x*2
# chamada a funcion
print('f(3)= ', f(3))
# definicion de f como unha funcion lambda (unha soa linha)
g = lambda x: x*2
# chamada a funcion g
print('g(3)= ', g(3))
```

- No canto da palabra reservada **`def`** utilízase **`lambda`**.
- Os argumentos de entrada non están entre paréntese.
- Non se utiliza a palabra reservada **`return`**.
- Donde se utilice unha función **`lambda`** pode utilizarse unha función normal.