CREACIÓN DE PROGRAMAS EN PYTHON

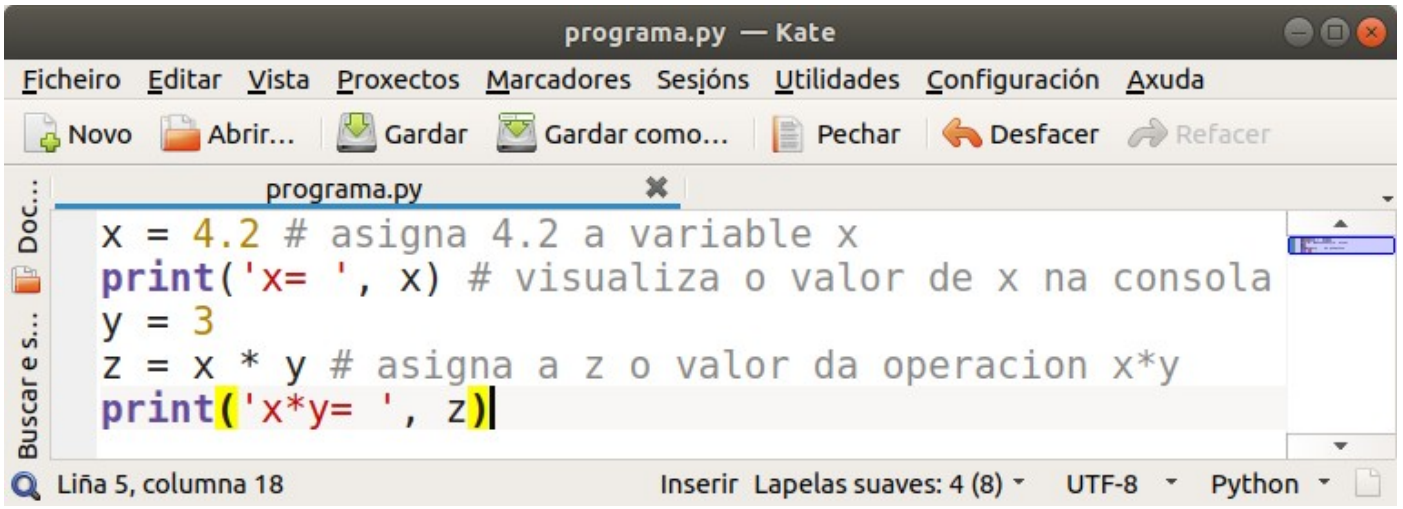
Grace Murray Hopper (1906-1992)

- Inventora de compiladores das linguaxes de programación A0 e B0, para o cálculo de nóminas
- Contra-almirante da US Navy
- Muller do ano en Informática (1969), primeira muller na British Computer Society (1973)

Scripts en Python

- A tradución literal sería guións, aínda que prefiro chamarlles **programas**.
- Un programa é un arquivo que contén código fonte en linguaxe Python.
- Os programas teñen extensión **`.py`**
- Para escribir os programas necesitas un editor de texto, mellor que teña resaltada a sintaxe de Python.
- Escribe un programa con comandos de Python, gardao con nome.py e executao desde o terminal de Linux con: **`python3 nome.py`**

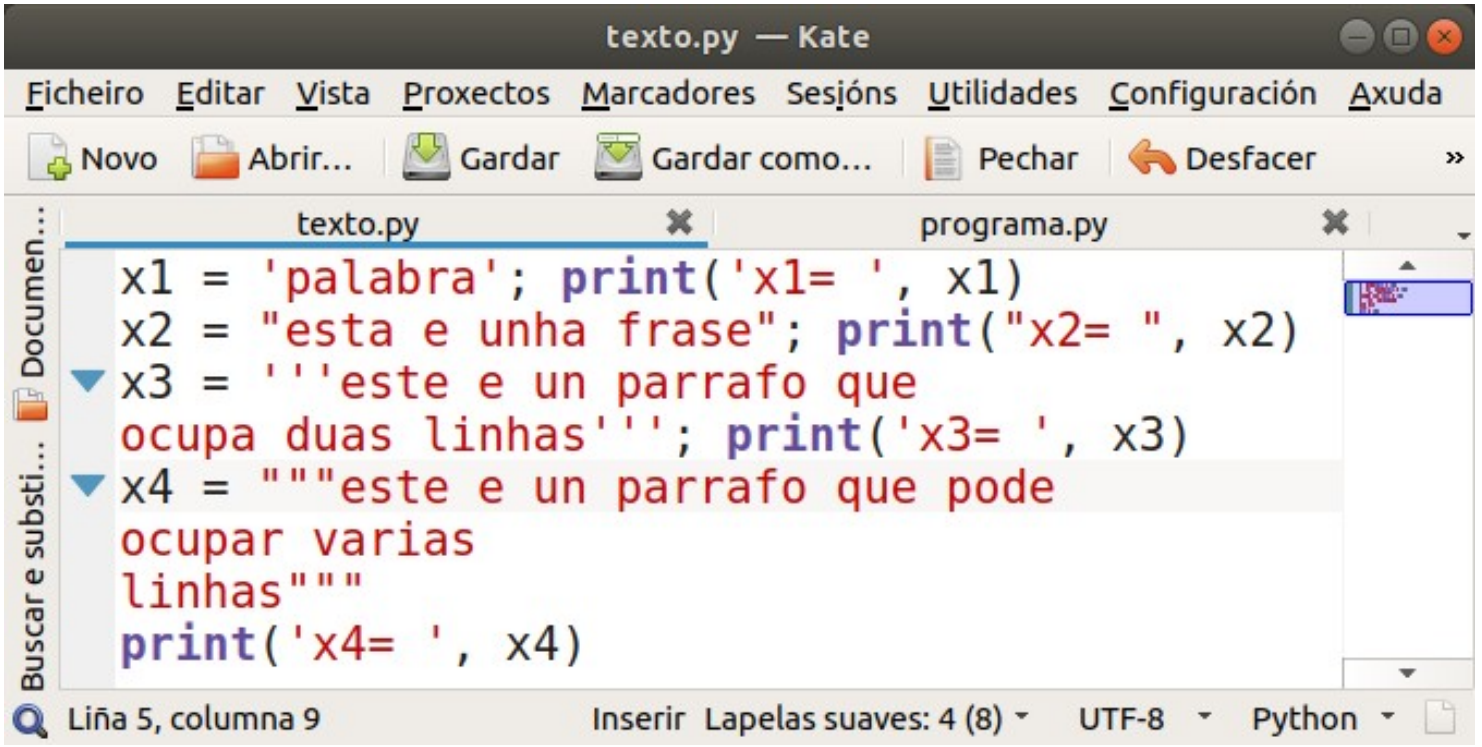
Exemplo de programa en Python



- As sentencias en Python rematan co cambio de liña. Nembargantes, Python permite continuar unha sentencia na seguinte liña co caracter **** .
  - x = 4.2 + \
  - 7.4 + \
  - 8.4
  - Equivalente a x=4.2 + 7.4 + 8.4
- Con sentencias que conteñan **[]**, {} ou **()** non se necesita usar o caracter de continuación de liña **** . Por exemplo, na definición dunha lista:
  - dias = [‘luns’, ‘martes’,
  - ‘mercores’, ‘xoves’]

Texto nos programas de Python

- Para poñer varias sentencias nunha liña, separalas por punto e coma (**;**).
- Python acepta comillas simples (‘), dobres (“) e triples (‘’’ ou “”””) para especificar o comezo e final dun texto literal ou cadea de caracteres. Exemplos:



Execución de Scripts en Windows

- Os arquivos `.py` xa están asociados ó interprete de Python, polo que con premer dúas veces sobre o arquivo xa se executa o programa.
- Os resultados da execución visualízanse na consola. Se a execución é moi rápida pode que non de tempo a vela.
- Truco: engadir ó arquivo co código unha liña que agarde a entrada dalgún dato por teclado: `input()`
- Así a consola agarda a que se introduza algo por teclado.

Máis sobre a execución de scripts en linux

- Executar un programa no terminal de Linux:

```
python3 nome.py
```

- Desde o entorno interactivo executar:

```
>>> run nome.py
```

- Execución implícita do arquivo nome.py:
  - Introducir na primeira liña do arquivo `#!/usr/bin/python3` (esta é a ruta onde está instalado python).
  - Darlle permisos de execución ó arquivo nome.py con:

```
chmod +x nome.py
```

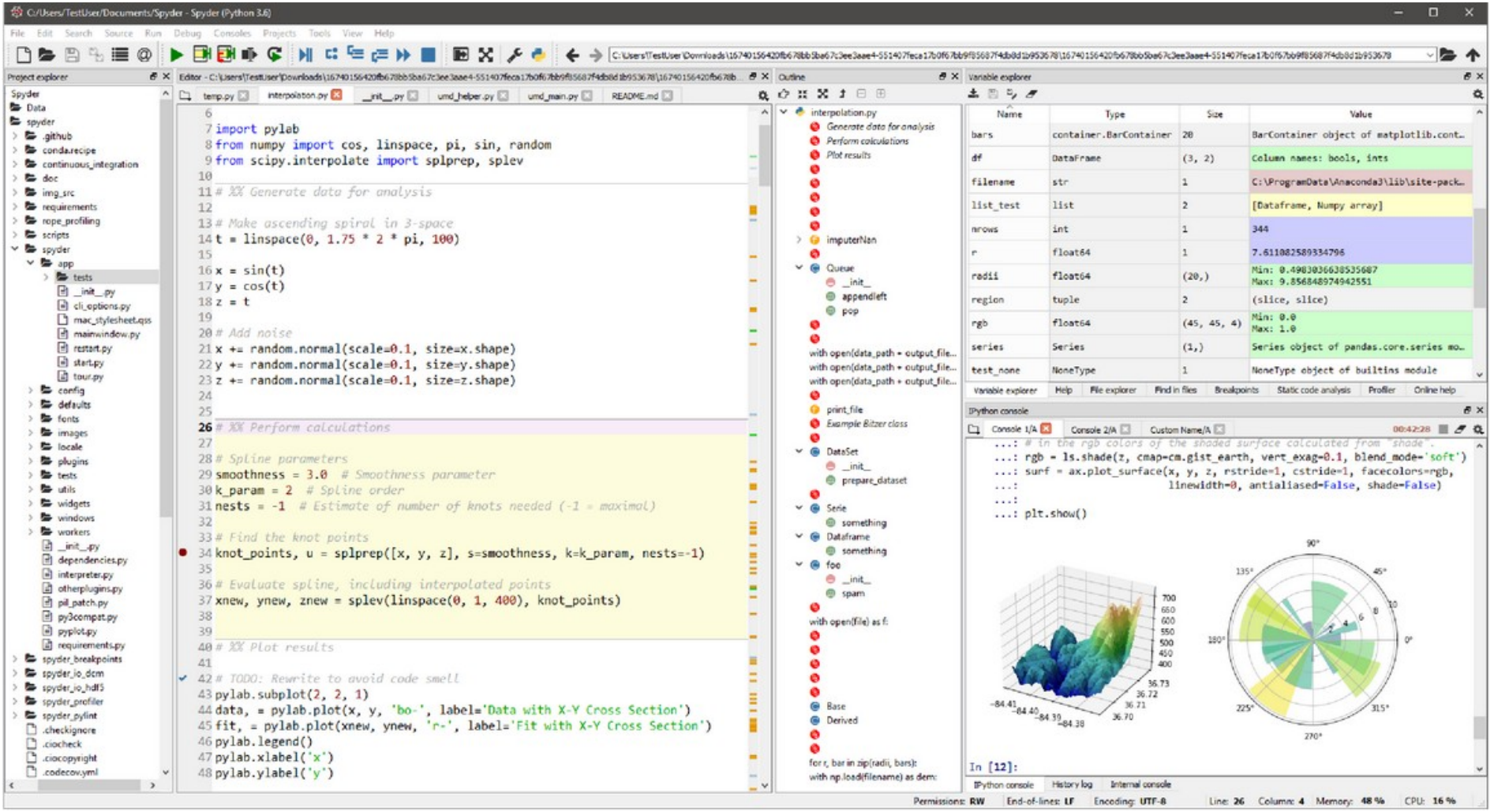
  - Executar o arquivo no terminal con:

```
./nome.py
```

Entornos de execución para Python

- Instalación dun entorno de desenvolvemento IDE (Interface Development Environment) no ordenador. O IDE máis popular é SPIDER (https://www.spyder-ide.org/).
- Execución de comandos ou programas online desde un navegador (non necesitar ter instalado Python no ordenador). Algúns exemplos podes atopalos no enlace: https://quintagroup.com/cms/python/online-interpreter.

Execución desde o entorno SPYDER



Execución desde o entorno SPYDER

- Desde o intérprete executar (como en Linux):

```
>>> run nome.py
```

- Pulsar o botón run na interface
- En ambos casos, necesítase arrancar o entorno para executar os programas. As opcións das primeiras transparencias permiten executar os programas sen entrar no entorno SPYDER.

Mensaxes de erro

- Ó executar un programa o resultado pode non ser correcto por:
  - **Erros na sintaxe de Python**: son máis ou menos doados de correxir a partires das trazas de rastreo que proporciona Python (liña e arquivo nos que se produce o erro). Máis información en: http://python.org.ar/MensajesExcepcionales
  - **Erros de execución**: o programa remata sen erros pero o resultado non é correcto. Utilizar o depurador.
  - Erros causados por excepcións non capturadas.

Entrada/saída de datos no programa

- Entrada de datos desde o teclado:
  - **`input("texto")`**: detén o programa, amosa na pantalla o texto que lle pasemos a función e agarda a que se introduza algo por teclado. A función devolve o que se introduce por teclado como unha cadea de caracteres que coas función **`int()`** ou **`float()`** podemos convertir a números.
- Saída de datos por pantalla:
  - **`print(*objects,sep='',end='\n',file=sys.stdout, flush=False):`** visualiza datos na consola. O argumento **objects** son os obxetos a visualizar, **sep** é o separador a utilizar entre os obxetos (espacio por defecto), **end** indica o que facer cando se executa **print()** (por defecto cambia de liña '\n'), se **flush** é True forza a que se visualice instantaneamente e **file** é un fluxo que por defecto representa a consola.
  - **`print():`** cambia de liña.

Saída de datos con formato

- Saída de datos con formato por pantalla:
  - **`print(cadea % (var1,var2,var3, ...))`**: onde cadea é unha cadea de caracteres con formatos intercalados, tantos como número de variables poñamos.
- Exemplos:
  - **`print('Visualiza enteiros x=%d e y=%d' % (34,40))`** # devolve Visualiza enteiros x=34 e y=40
  - **`print("Visualiza reais x=%7.2f e y=%.3f" % (45.4321, 40.2))`** # Visualiza reais x= 45.43 e y=40.200
  - **`print('Hola %s' % ('Python'))`** # devolve Hola Python
- Formatos: **%d** (enteiro), **%f** (real), **%e** (real en formato exponencial), **%g** (real, elixe entre **%f** e **%e**), **%s** (cadea de caracteres), **%c** (caracter).
- Formatos: **%m.nf** (**m** significa o ancho do campo e **n** o número de decimais).

Entrada/saída: arquivos de texto

- Pasos a realizar: abrir arquivo, ler/escribir, cerrar arquivo.
- Abrir arquivo: **`f=open(nome_arquivo, modo)`**
  - nome_arquivo é a ruta do arquivo a ler/escribir.
  - Modo especifica o uso que se vai facer do arquivo: **`'r'`** (so para lectura), **`'w'`** (so para escribir desde o principio do arquivo), **`'a'`** (para escribir ó final dun arquivo) e **`'r+'`** (para ler e escribir no arquivo).
  - f é o obxecto de tipo **file** devolto por **fopen**.
- Pechar arquivo: **`f.close()`**
- Métodos do obxecto file:
  - **`f.read(tamanho)`**: devolve unha cadea de caracteres cos tamanho bytes do arquivo f. Se non especificas tamanho le todo o arquivo.
  - **`f.readline()`**: devolve unha liña do arquivo (ate que atope o carácter fin de liña (**`'\n'`**).

Entrada/saída: arquivos de texto

- Métodos do obxecto **file**:
  - **`f.write(cadea)`**: escribe unha cadea de caracteres no arquivo.
  - **`f.writelines(secuencia)`**: escribe unha secuencia de cadeas de caracteres no arquivo (cada elemento da secuencia nunha liña).
  - **`f.tell()`**: devolve un enteiro coa posición actual de lectura/escritura no arquivo.
  - **`f.seek(bytes)`**: move a posición actual de lectura/escritura a bytes desde o inicio do arquivo.