Vocabulario de PYTHON

Algúns trucos iniciais:

- Na aula, todas as persoas temos un nome. Pois nos ordenadores, cando construimos programas, tamén temos que poñerlle un nome a todos os datos. Pero, con moito tino.... na aula se dúas persoas teñen o mesmo nome sempre hai confusión e, normalmente, usamos o seu apelido para desfacer a confusión. Tamén nos programas temos que evitar poñer o mesmo nome a distintos datos. Dáselle un valor a ese nome co signo "=", por exemplo suma=20 ou tempo=3.
- Os nosos coñecementos do inglés serviránnos a hora de aprender as linguaxes de programación, xa que éstas foron inventadas nos paises anglosaxóns e teñen moitos préstamos da língua inglesa. Por este motivo, non uses acentos nos contos dos ordenadores.
- Para construir programas necesitamos, además de coñecer a linguaxe de programación, ter a axuda dun traductor, outro programa que traduce o meu conto (na linguaxe de programación) para que o entenda o electródoméstico de plástico con seus compoñentes electrónicos. No caso de PYTHON chámaselle a este traductor **o intérprete de python**.
- Imos utilizar na actividade un interprete de python en internet: https://www.online-python.com Se o interprete de python en internet comeza a poñer publicidade molesta, busca outro intérprete en liña en internet. No panel superior escribimos o programa (conto na linguaxe de programación python) e cando rematemos prememos no botón **Run** para visualizar no panel inferior a execución do programa. O panel inferior (despois de >) podemos usalo como unha calculadora básica para facer operacións aritméticas. Exemplo: 4+5 (suma), 7-3 (resta), 4*5 (multiplicación) e 6/2 (división). Despois de poñer cada operación, preme a tecla intro para executar.
- Todo o que está despois do símbolo # e que aparece en verde nas dúas primeiras liñas da figura 5 non son considerados polo intérprete de python (compiler), é dicir, son comentarios. Este programa por defecto ó conectarnos a páxina web so ten unha frase (sentencia) ou liña "**print**(**"Hello world"**)". Se prememos o botón **Run,** visualiza en pantalla (función **print()**) o texto que está entre comillas (Hello world).

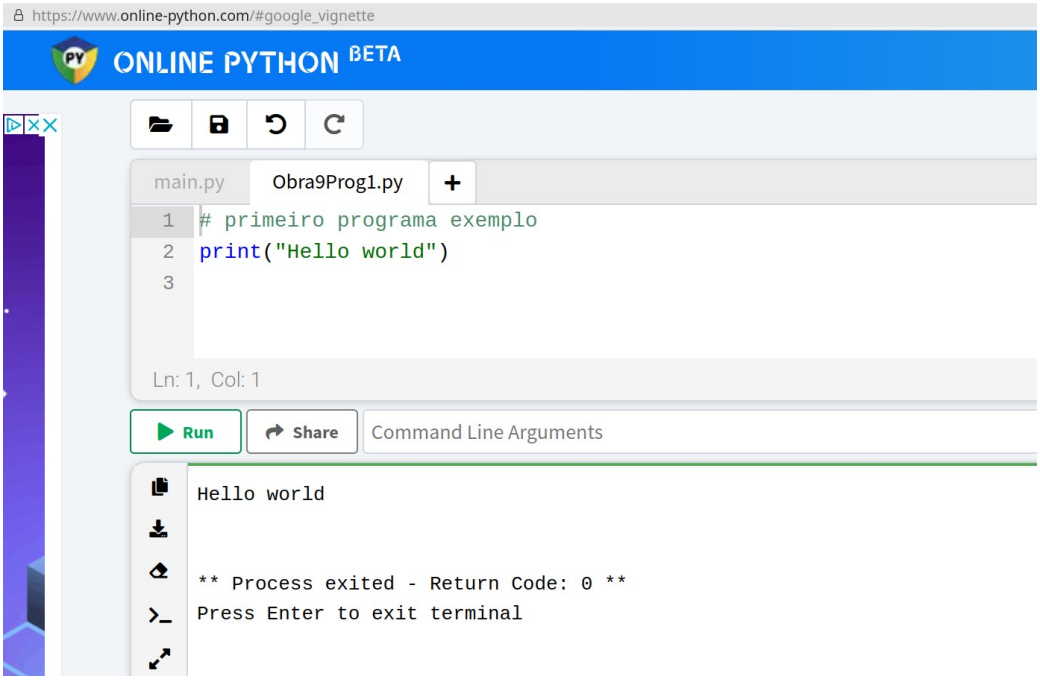

Figure 5: Captura de pantalla do intérprete de Python en internet executando o programa por defecto.

Copia o programa Obra14Prog1.py:

```
# primeiro programa exemplo
print("Hello world")
```

- **O interprete de python non permite ningunha falta de ortografía**. Con algunha falta de ortografía, o programa da erros e non nos permite executalo (por exemplo, o texto que aparece na pantalla vai entre comillas “”, que non escribimos ben a palabra **print**, que nos esquecemos un paréntese, etc.

O exemplo anterior é un exemplo trivial que so nos da unha benvida. Na práctica, a maioría dos programas piden algún dato por teclado (python proporciona a función **input()**) e amosan os resultados da execución do programa na pantalla do ordenador (función **print()** de python). Porque a nosa intención ó deseñar programas é que fagan unha tarefa, pero que en execución distintas poida introducir distintos datos e observar distintos resultados.

O conto de ordenador máis simple sería un conto infantil, un conto que ten poucas frases (sentencias) e que sexan simples. Por exemplo, **facer un programa que conteña a lista de estudantes de clase cos nomes ordeados alfabéticamente e que pida por teclado un número para devolver o nome do ou da estudante nesa posición da lista** (supón que se introduce un

número válido). O programa sería o mostrado na figura 6: 1) a liña 1 define estudantes como a lista de nomes de estudantes da clase ordeados alfabéticamente, fixade que a lista son elementos entre corchetes separados por , e con cada nome entre comillas; 2) a liña 2 pide por teclado datos coa función *input()*, pero o que lemos con esta función sempre é texto, polo que hai que convertilo a un número coa función *int()*; 3) a liña 3 usa a función print() para amosar o resultado na pantalla (panel dereito da figura) que é a frase Estudante seguido do contido de estudantes[numero -1]. Estudantes é unha lista de nomes e eu quero ver o valor da posición "numero" da lista (poñemos numero-1 porque o python comeza a contar en 0 no canto de 1, algo moi pouco intuitivo). Agora queda executar o programa premendo no botón **Run** e aparece a mensaxe "Número da lista: " e o cursor queda ahí esperando que introduzamos ese dato e premamos a tecla Intro, ó facelo o programa continua a súa execución ate rematar, como se observa na figura 6.

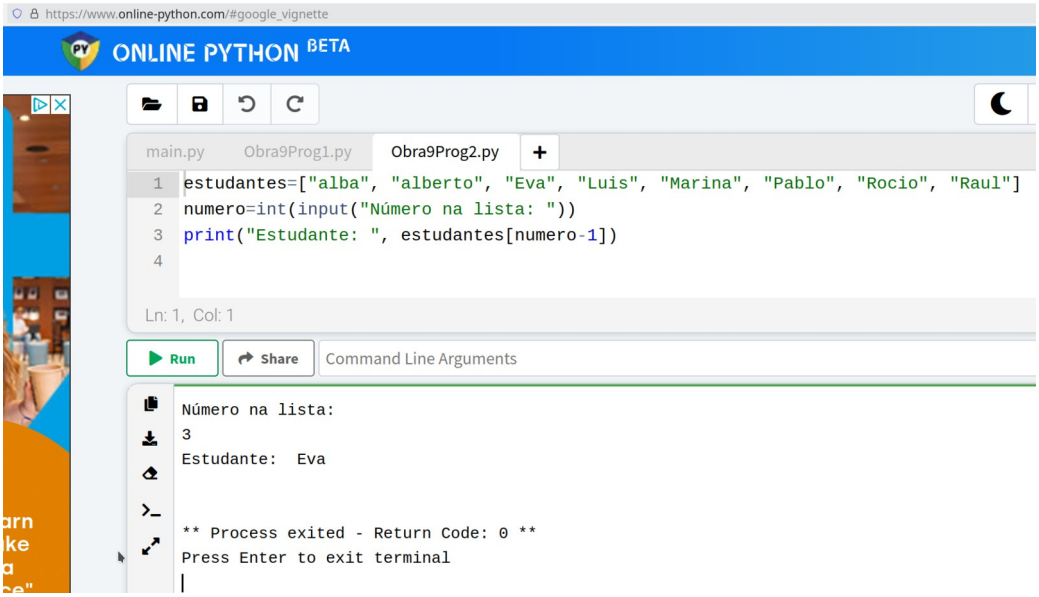

Figure 6: Exemplo de execución do programa da lista da clase.

Copia o programa Obra14Prog2.py:

```
estudantes=["alba", "alberto", "Eva", "Luis", "Marina", "Pablo",
"Rocio", "Raul"]

numero=int(input("Número na lista: "))
print("Estudante: ", estudantes[numero-1])
```

Imos continuar aprendendo a construir programas en Python. O alumnado preguntarase ...**¿é cómo construimos esas bifurcacións e repeticións?**. Para iso temos que aprender cales son as palabras reservadas da linguaxe de programación Python (o vocabulario) para realizar eses mecanismos:

- **Bifurcacións**: utiliza as palabras resevadas **if**, **elif**, **else** . Despois do **if** avalíase unha condición (por exemplo, que un dato sexa maior ou igual a unha constante ou outras condicións cotiás). Se a condición se cumpre, executánse o bloque de sentencias (frases)

que están a continuación e, se non se cumpre a condición, executanse as sentencias que están despois do **else**. Teño visto algúns libros infantís que te invitan a resolver pequenas probas ou acertixos e, en función do resultado, te din que saltes a unha páxina ou outra.

- **Repeticións definidas**: (as que sabemos o número de veces que se van a repetir antes de comezar as repeticións). Realízanse coa palabra resevada **for** e xa veremos como lle decimos o número de veces a repetir o bloque de sentencias (frases) que ven debaixo do **for**.
- **Repeticións indefinidas**: (as que non sabemos a priori o número de veces que se van a repetir e só o podemos saber despois de que rematen as repeticións). Utilizan a palabra reservada **while**. Despois do **while** avalian unha condición, e se se cumpre, repítese o bloque de sentencias ou código que está a continuación. Despois volve a avaliarse a condición que está despois do **while** e se se cumpre volve a repetirse o bloque de sentencias, e así sucesivamente mentres que se siga cumprindo a condición.

Imos escenificar estes funcionamentos con exemplos matemáticos simples que o alumnado facía no papel cando estaba en primeiro ou segundo de educación primaria. Sempre lembrádelle ó alumnado que os ordenadores son como bebes ós que temos que ensinar as cousas desde cero, porque non saben absolutamente nada. Entón, podes plantexar estes exercicios como se foran mestras e mestres que están ensinando matemáticas ó ordenador (como a un bebé). Algúns exemplos serían:

Facer un programa utilizando a linguaxe de programación Python que construa unha secuencia de números crecentes entre 0 e 40 de 3 en 3.

O programa (tamén chamado algunhas veces código fonte) está na parte superior da figura 7 e o resultado da súa execución está na parte inferior. No código fonte, a primeira liña é un comentario explicando o que fai o programa, que non vai ter en conta o intérprete de python. Cada liña do programa é unha sentencia, que sería o equivalente a unha frase nun conto. Da liña 2 á 4 son liñas que asignan un valor a un dato con nome: o valor inicial da secuencia ó dato con nome “inicio” na liña 2, o valor final da secuencia ó dato que chamamos “fin” na liña 3 e, finalmente, o incremento para construir a secuencia chamámoslle “incremento” e poñémoslle o valor 3 na liña 4. Con isto xa temos as personaxes do noso conto e, a cada un, puxémoslle un nome. Para construir a secuencia, non sei a priori os números que vai ter a miña secuencia dun xeito simple (sen facer cálculos) polo que debería de utilizar unha repetición indefinida. Eso fai a liña 5 que uso a palabra reservada **while** seguida da condición “inicio < fin”. Vedes que a frase completa é: **while inicio < fin :** (os : do final son parte do vocabulario de python e indica o comezo do bloque de sentencias que se ten que repetir. Resulta pouco intuitivo, pero moitas veces cando estudamos outros idiomas convencionais tamén existe algunha regra pouco intuitiva). O bloque de sentencias que se ten que repetir son as liñas 6 e 7 (que teñen que comezar cunha tabulación despois da primeira columna (tamén é unha regra da linguaxe python) --- tes que pulsar a tecla con frecha que está na parte esquerda do teclado, normalmente o lado da Q---. Repara en que, neste editor da linguaxe de programación Python, todas as palabras reservadas, aparecen en cor azul (cada editor ter a súa configuración por defecto). **¿E cales son as tarefas (sentencias ou frases) que se repiten?** Pois a liña 6 simplemente visualiza na pantalla o valor do dato “inicio”, coa función **print()** como xa comentamos. A liña 7 actualiza o valor do dato “inicio” sumándolle o dato “incremento”. Esta forma de actualizar os datos na programación é un concepto elemental para programas moito máis complexos, pero que tamén reflicte o noso comportamento na vida real. Imaxina que imos a un souto a recoller castañas e levamos unha bolsa (baleira, por suposto). Esa bolsa baleira sería o noso dato “inicio” xa que contén 0 castañas. Andamos polo souto e recollemos 2 castañas que metemos na bolsa.... ¿Cántas castañas teño agora? Terei as que xa tiña (que eran 0) + as que botei na bolsa (que foron 2). Polo tanto, inicio = 0 + 2=2. Collemos outras 3 castañas e metémolas na bolsa. ¿Cantas teño agora? Terei 5 que son o resultado de sumar as que tiña máis as 3 que metín, é dicir, inicio = 2+3=5, e así sucesivamente.

Unha vez que se executa a liña 7, o programa volve ó inicio da repetición (liña 5) e volve a comprobar a condición, e así sucesivamente mentres que se siga cumprindo a condición. Cando a condición deixa de cumprirse, o programa remata. Se pulsamos o botón **Run** na figura 7, obtense o resultado do panel inferior desta figura. O último número é 39, porque despois de visualizar 39, faríamos inicio = inicio + incremento = 39 + 3 = 42, pero 42 non cumpre a condición de ser maior que fin (que vale 40) e o programa remata.

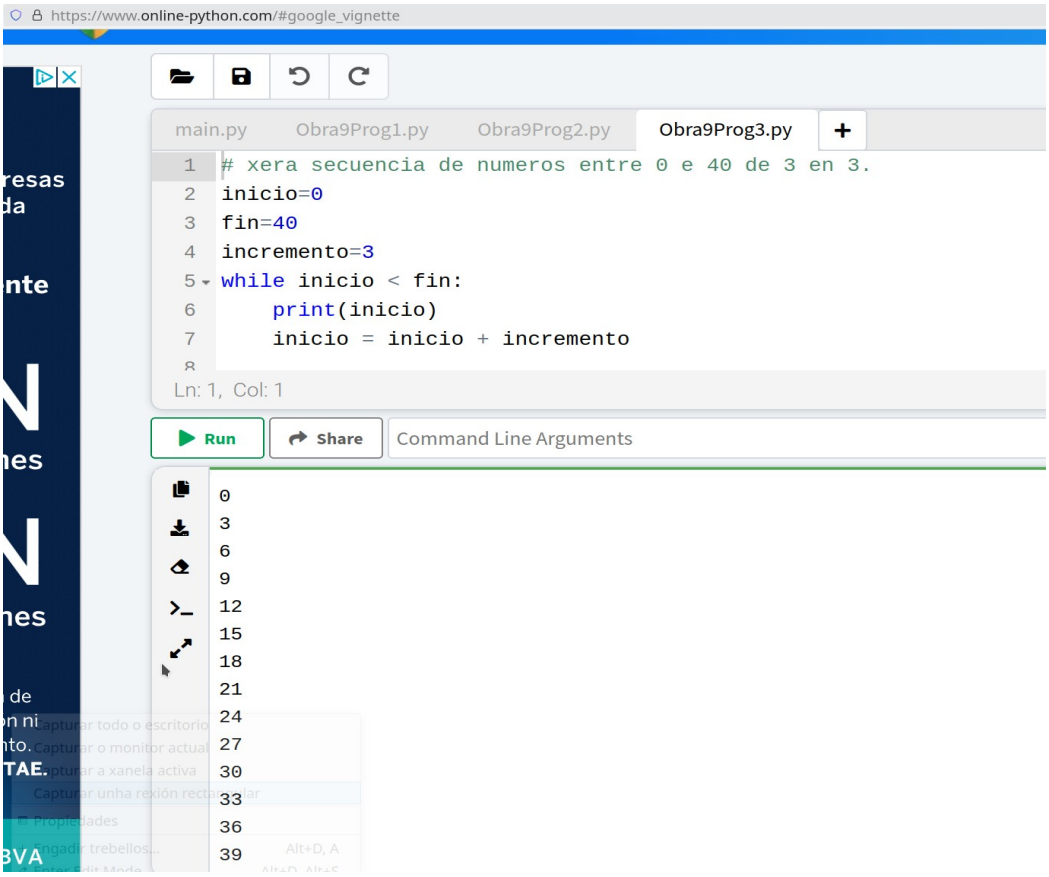

Figure 7: Exemplo da secuencia de números crecentes de 3 en 3 entre o 0 e o 40

Copia o programa Obra14Prog3.py:

```
# xera secuencia de numeros entre 0 e 40 de 3 en 3.
inicio=0
fin=40
incremento=3
while inicio < fin:
  print(inicio)
  inicio = inicio + incremento
```