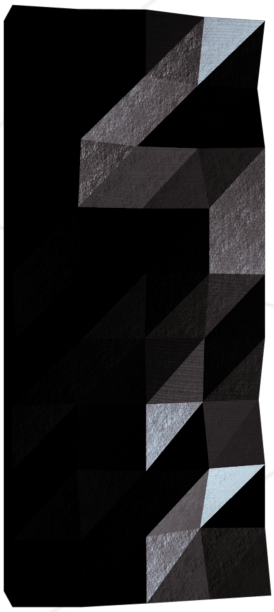

Ruby - Unidad 05

Tablas y funciones asociadas

Staff Pedagógico 42 pedago@42.es

Resumen: En esta unidad, vamos a ver cómo utilizar las tablas y las funciones asociadas.

Índice general

Capítulo I

Preámbulo

`Receta de la pissaladière`

Preparación
TIEMPO TOTAL: 1H15
Preparación: 30 min
Cocción: 45 min

Etapa 1
Calentamos 5 cucharadas soperas de aceite de oliva en una sartén profunda.

Etapa 2
Cortamos las cebollas en rodajas.

Etapa 3
Cuando el aceite esté caliente echamos las cebollas en la sartén, añadimos pimienta, las hierbas aromáticas y el azúcar. Sobre todo no añadimos sal porque ya tenemos las anchoas. El azúcar es indispensable para quitar el acidez de la cebolla.

Etapa 4
Sofreímos las cebollas hasta que se pongan ligeramente amarillas. No tenemos que dorarlas demasiado ya que van a seguir cocinándose en el horno.

Etapa 5
El secreto de la Pissaladière está aquí. Tenemos que conservar algunas anchoas y echar el resto en la sartén con las cebollas. Las anchoas se van a derretir con el calor y a mezclarse con las cebollas. Si podemos añadirle una cucharada sopera del aceite de las anchoas a la preparación quedará aún mejor.

Etapa 6
Extendemos la masa de pan sobre la bandeja del horno que habremos embadurnado previamente con aceite de oliva.

Etapa 7
Vertemos la preparación sobre la masa y decoramos con anchoas y aceitunas.

Etapa 8
Precalentamos el horno a 220°C y a continuación metemos la pissaladière.

Etapa 9
Respecto al tiempo de cocción, en cuanto veamos que la masa de pan esté cocida (miramos los bordes de la masa) podemos sacar la pissaladière.

Etapa 10
Tenemos que dejar que se enfríe la pissaladière, dado que se degusta fría.

Etapa 11
La podemos acompañar con una ensalada de escarola, y con un vino rosado de Bandol está exquisito.

Capítulo II

Instrucciones Generales

Salvo que se indique lo contrario de forma explícita, las siguientes instrucciones serán válidas durante el tiempo que dure la Piscina.

- Esta evaluación será la única referencia: no se fíe de los rumores de pasillo.
- Los ejercicios han sido ordenados con mucha precisión del más sencillo al más complejo. En ningún caso prestaremos atención a un ejercicio complejo ni lo tendremos en cuenta si no se ha conseguido realizar perfectamente un ejercicio más sencillo.
- Tenga cuidado con los permisos de sus archivos y de sus directorios.
- Sus compañeros de Piscina se encargarán de corregir los ejercicios que usted realice.
- Para la evaluación entre pares, no debe dejar en su directorio ningún archivo que no haya sido indicado de forma explícita en los enunciados de los ejercicios.
- ¿Tiene alguna pregunta? Pregunte a su vecino de la derecha. Si no, pruebe con su vecino de la izquierda.
- Todas las respuestas a sus preguntas técnicas se encuentran en los `man` o en Internet.
- ¡No olvide participar en el foro Piscina de su Intranet y en el Slack!
- Lea detenidamente los ejemplos ya que pueden ayudarle a identificar algún trabajo que tenga que realizar y que, a primera vista, no venga explicado en el enunciado.
- Razone. ¡Se lo suplico, por Thor, por Odín!

Capítulo III

Ejercicio 00: create_array

	Ejercicio : 00
Crear un array	
Directorio de entrega : $ex00/$	
Ficheros a entregar : `create_array.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `create_array.rb`.
- Tiene que ser un programa ejecutable.
- Deberá definir un array de números.
- Mostrará el array en la pantalla:

```
?> ./create_array.rb | cat -e
[2, 8, 9, 48, 8, 22, -12, 2]$
?>
```

`print, puts, p.`

Capítulo IV

Ejercicio 01: play_with_arrays

	Ejercicio : 01
Manipular arrays	
Directorio de entrega : $ex01/$	
Ficheros a entregar : `play_with_arrays.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `play_with_arrays.rb`.
- Tiene que ser un programa ejecutable.
- Primero tendrá que definir un array de números.
- Después, el programa deberá hacer una iteración sobre ese array y crear un array nuevo añadiéndole 2 a cada valor del array original.
- Por lo tanto, tiene que haber dos arrays en el programa, el original y el que haya creado nuevo.
- Para terminar, muestre los dos arrays en la pantalla utilizando el método p en lugar de puts.
- Por ejemplo, si el array original es [2, 8, 9, 48, 8, 22, -12, 2], obtendrá la siguiente salida:

```
?> ./play_with_arrays.rb | cat -e
[2, 8, 9, 48, 8, 22, -12, 2]$
[4, 10, 11, 50, 10, 24, -10, 4]$
?>
```

`Google "método p en ruby", each`

Capítulo V

Ejercicio 02: play_with_arrays++

	Ejercicio : 02
Manipular arrays, otra vez	
Directorio de entrega : $ex02/$	
Ficheros a entregar : `play_with_arrays.rb`	
Funciones autorizadas : `Todas`	

- Recupere el programa anterior, pero esta vez trate únicamente los valores del array original superiores a 5.
- Por ejemplo, si el array original es [2, 8, 9, 48, 8, 22, -12, 2], obtendrá la siguiente salida:

```
?> ./play_with_arrays.rb | cat -e
[2, 8, 9, 48, 8, 22, -12, 2]$
[10, 11, 50, 10, 24]$
?>
```

Capítulo VI

Ejercicio 03: play_with_arrays+=2

	Ejercicio : 03
Manipular arrays, ¡otra vez más!	
Directorio de entrega : $ex03/$	
Ficheros a entregar : `play_with_arrays.rb`	
Funciones autorizadas : `Todas`	

- Recupere el programa anterior, pero esta vez no muestre en la salida los valores que se repitan. Cuidado, no debe retirar de forma explícita valores de los arrays.
- Por ejemplo, si el array original es [2, 8, 9, 48, 8, 22, -12, 2], obtendrá la siguiente salida:

```
?> ./play_with_arrays.rb | cat -e
[2, 8, 9, 48, 22, -12]$
[10, 11, 50, 24]$
?>
```

`Uniq`.

Capítulo VII

Ejercicio 04: Parámetros

	Ejercicio : 04
Vale, ¿pero cuántos?	
Directorio de entrega : $ex04/$	
Ficheros a entregar : `parameters.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `parameters.rb`.
- Tiene que ser un programa ejecutable.
- El programa mostrará el número de parámetros que haya recibido, seguido de un salto de línea.

```
?> ./parameters.rb
Número de parámetros : 0.
?> ./parameters.rb "initiación"
Número de parámetros : 1.
?> ./parameters "qué" "locura" "hay" "por" "doquier!"
Número de parámetros : 5.
?>
```

`Google ARGV, array size.`

Capítulo VIII

Ejercicio 05: most_first_param

	Ejercicio : 05
Mostrar un parámetro	
Directorio de entrega : $ex05/$	
Ficheros a entregar : `most_first_param.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `most_first_param.rb`.
- Tiene que ser un programa ejecutable.
- El programa mostrará la primera cadena de caracteres pasada como parámetro, seguida de un salto de línea.
- Si no hay ningún parámetro, mostrará `none` seguido de un salto de línea.

```
?> ./aff_first_param.rb | cat -e
none$
?> ./aff_first_param.rb "Code Ninja" "Numerique" "42" | cat -e
Code Ninja$
?>
```

Busque cómo se utilizan las sentencias condicionales if.

Capítulo IX

Ejercicio 06: UPCASE_IT

	Ejercicio : 06
Ver más grande	
Directorio de entrega : $ex06/$	
Ficheros a entregar : `upcase_it.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `upcase_it.rb` que reciba como parámetro una cadena de caracteres.
- Tiene que ser un programa ejecutable.
- El programa mostrará la cadena de caracteres en mayúsculas seguida de un salto de línea.
- Si el número de parámetros es distinto de 1, mostrará `none` seguido de un salto de línea.

```
?> ./upcase_it.rb | cat -e
none$
?> ./upcase_it.rb "iniciación" | cat -e
INITIATION$
?> ./upcase_it.rb 'EsTe EjErCiCiO eS bAsTaNtE FáCiL !' | cat -e
ESTE EJERCICIO ES BASTANTE FÁCIL !$
?>
```

Capítulo X

Ejercicio 07: downcase_it

	Ejercicio : 07
Ver más pequeño	
Directorio de entrega : $ex07/$	
Ficheros a entregar : `downcase_it.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `downcase_it.rb` que reciba como parámetro una cadena de caracteres.
- Tiene que ser un programa ejecutable.
- El programa mostrará la cadena de caracteres en minúsculas seguida de un salto de línea.
- Si el número de parámetros es distinto de 1, mostrará `none` seguido de un salto de línea.

```
?> ./downcase_it.rb | cat -e
none$
?> ./downcase_it.rb "LUCIOLE" | cat -e
luciole$
?> ./downcase_it.rb 'EsTe EjErCiCiO eS bAsTaNtE FáCiL !' | cat -e
este ejercicio es bastante fácil !$
?>
```

`Este ejercicio no debería llevarle más de 10 segundos.`

Capítulo XI

Ejercicio 08: most_rev_params

	Ejercicio : 08
Mostrar una tabla al revés	
Directorio de entrega : $ex08/$	
Ficheros a entregar : `most_rev_params.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `most_rev_params.rb`.
- Tiene que ser un programa ejecutable.
- Cuando se ejecute el programa mostrará todas la cadenas de caracteres pasadas como parámetros, seguidas de un salto de línea y en orden inverso.
- Si hay menos de dos parámetros, mostrará `none` seguido de un salto de línea.

```
?> ./aff_rev_params.rb | cat -e
none$
?> ./aff_rev_params.rb "hola" | cat -e
none$
?> ./aff_rev_params.rb "Ruby" "piscina" "hola a la" | cat -e
hola a la$
piscina$
Ruby$
?>
```

`Google array reverse.`

Capítulo XII

Ejercicio 09: scan_it

	Ejercicio : 09
Escanear un texto	
Directorio de entrega : $ex09/$	
Ficheros a entregar : `scan_it.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `scan_it.rb` que reciba dos parámetros.
- El primer parámetro será una palabra clave que tendrá que buscar en una cadena.
- El segundo parámetro será la cadena que deberá recorrer.
- Tiene que ser un programa ejecutable.
- Cuando se ejecute el programa mostrará el número de veces que se encuentra la palabra clave en la cadena.
- Si el número de parámetros es distinto de 2 o si la primera cadena no aparece en la segunda, mostrará `none` seguido de un salto de línea.

```
?> ./scan_it.rb | cat -e
none\$
?> ./scan_it.rb "los" | cat -e
none\$
?> ./scan_it.rb "los" "los ejercicios del C05 no son los más difíciles" | cat -e
3\$ ?>
```

`Google "Ruby scan method".`

Capítulo XIII

Ejercicio 10: comparación

	Ejercicio : 10
Identificar un parámetro	
Directorio de entrega : $ex10/$	
Ficheros a entregar : `parameter_matching.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `parameter_matching.rb`.
- Tiene que ser un programa ejecutable.
- Si se le pasa un parámetro como argumento, el programa tendrá que pedirle al usuario que introduzca una palabra.
- Si la palabra introducida por el usuario es la misma que la que se ha pasado como parámetro, el programa mostrará "Sí Señor!", si no mostrará "No, se siente... seguido de un salto de línea.
- Si el número de parámetros pasados al programa es distinto de 1, mostrará `none` seguido de un salto de línea.

```
?> ./parameter_matching.rb
none
?> ./parameter_matching.rb "Hello"
Cuál era el parámetro ? Hola
No, se siente...
?> ./parameter_matching.rb "Hello"
Cuál era el parámetro ? Hello
Sí Señor!
?>
```

`Utilice una (¿o varias? :) ) estructura if ... else ...`

Capítulo XIV

Ejercicio 11: count_it

	Ejercicio : 11
Contar y medir parámetros	
Directorio de entrega : $ex11/$	
Ficheros a entregar : `count_it.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `count_it.rb`.
- Tiene que ser un programa ejecutable.
- El programa mostrará `"parametros:"`, a continuación el número de parámetros pasados como argumentos seguido de un salto de línea, después cada parámetro y su tamaño seguido de un salto de línea.
- Si no hay ningún parámetro, mostrará `none` seguido de un salto de línea.

```
?> ./count_it.rb | cat -e
none$
?> ./count_it.rb "Game" "of" "Thrones" | cat -e
parametres: 3$
Game: 4$
of: 2$
Thrones: 7$
?>
```

Esta vez, utilice el método each en lugar de los bucles while.

Capítulo XV

Ejercicio 12: string_are_arrays

	Ejercicio : 12
Comprender mejor las cadenas de caracteres	
Directorio de entrega : $ex12/$	
Ficheros a entregar : `string_are_arrays.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `string_are_arrays.rb` que reciba como parámetro una cadena de caracteres.
- Tiene que ser un programa ejecutable.
- Cuando se ejecute el programa mostrará `"z"` para cada carácter `"z"` que aparezca en la cadena pasada como parámetro, todo ello seguido de un salto de línea.
- Si el número de parámetros es distinto de `1` o si no hay ningún carácter `"z"` en la cadena, mostrará `none` seguido de un salto de línea.

```
?> ./strings_are_arrays.rb | cat -e
none$
?> ./strings_are_arrays.rb "El carácter buscado no se encuentra en esta cadena de
    caracteres de caracteres" | cat -e
none$
?> ./strings_are_arrays.rb "z" | cat -e
z$
?> ./strings_are_arrays.rb "Zebulón se calzó unos zapatos para cazar en la zarzamora" |
    cat -e
zzzzzz$ ?>
```

Las cadenas de caracteres también están compuestas por casillas, como los arrays. ¡Pruebe!

Capítulo XVI

Ejercicio 13: append_it

	Ejercicio : 13
Modificar cadenas de caracteres	
Directorio de entrega : $ex13/$	
Ficheros a entregar : `append_it.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `append_it.rb`.
- Tiene que ser un programa ejecutable.
- El programa mostrará los parámetros pasados como argumento, uno a uno, retirando la última letra de cada parámetro y mostrando ïsmo.[en] su lugar.
- Si el parámetro ya termina en ïsmo", no lo mostrará y pasará al siguiente.
- Si no hay ningún parámetro, mostrará `none` seguido de un salto de línea.

```
?> ./append_it.rb | cat -e
none$
?> ./append_it.rb "paralelo" "egoismo" "morale" | cat -e
paralelismo$
moralismo$
?>
```

`Match.`

Capítulo XVII

Ejercicio 14: free_range

	Ejercicio : 14
¡De aquí a allí, solo hay un paso!	
Directorio de entrega : $ex14/$	
Ficheros a entregar : `free_range.rb`	
Funciones autorizadas : `Todas`	

- Cree el programa `free_range.rb` que reciba dos parámetros.
- Esos dos parámetros serán dos números.
- Tiene que ser un programa ejecutable.
- El programa tendrá que construir un array que contenga todos los valores que existan entre esos dos números utilizando un rango. Después, mostrará el array con el método p.
- Si el número de parámetros es distinto de 2, mostrará `'none'` seguido de un salto de línea.

```
?> ./free_range.rb | cat -e
none$
?> ./free_range.rb 10 14 | cat -e
[10, 11, 12, 13, 14]$
?>
```

`Google range.`