## Exo 1

$ ruby exo\_01.rb

Puts « bonjour, monde ! »

## Exo 2

$ ruby exo\_02.rb

puts « Bonjour, monde ! »

puts « Et avec une voix sexy, ça donne : Bonjour, monde »

puts= est automatiquement suivit d’un saut à la ligne

print = arrive directement après le texte précédent sans saut à la ligne.

## Exo 3

exo\_03.rb

si on commence une ligne par « # » celle-ci ne sera pas lu et considéré comme un commentaire.

## Exo 4

exo\_04.rb

puts "Salut, ça farte ?

Le programme du dessus ne fonctionne pas car il manque “ »“ à la fin de la ligne. Le code une fois corrigé serait donc :

puts "Salut, ça farte ?**»**

## Exo 5

La partie « #{} » nous indique qu’il y aura un calcul sur les éléments entre les deux parenthèse. Et que le résultat de ce calcul sera inséré dans la phrase lorsqu’elle sera affichée.

# exo\_05.rb

**# on va afficher un texte**

puts "On va compter le nombre d'heures de travail à THP"

**# le calcul sera fait ce qui affichera le message « travail : 550 »**

puts "Travail : #{10 \* 5 \* 11}"

**# le calcul sera fait ce qui affichera le message « En minutes ça fait :33 000 »**

puts "En minutes ça fait : #{10 \* 5 \* 11 \* 60}"

**# on va afficher le texte « Et en secondes ? »**

puts "Et en secondes ?"

**# on vas affiche le résultat du calcul soit « 1 980 000 »**

puts 10 \* 5 \* 11 \* 60 \* 60

**# nous n’avons pas la partie «#{} » donc le calcul ne sera pas réalisé et le texte entre parenthèse sera affiché sans modification.**

puts "Est-ce que c'est vrai que 3 + 2 < 5 - 7 ?"

**#on affiche le résultat du teste de véracité de « 3 + 2 < 5 – 7 » soit**

**« 5 < -2 » qui sera donc « false »**

puts 3 + 2 < 5 - 7

**# Il y aura un calcul dans la phrase, ce qui affichera :**

**« Ça fait combien 3 + 2 ? 5 »**

puts "Ça fait combien 3 + 2 ? #{3 + 2}"

**# Il y aura un calcul dans la phrase, ce qui affichera :**

**« Ça fait combien 5 - 7 ? -2 »**

puts "Ça fait combien 5 - 7 ? #{5 - 7}"

**# Le texte « Ok, c'est faux alors ! » sera affiché.**

puts "Ok, c'est faux alors !"

**# Le texte « C'est drôle ça, faisons-en plus :» sera affiché.**

puts "C'est drôle ça, faisons-en plus :"

**# Il y aura un test de véracité dans la phrase, ce qui affichera :**

**« Est-ce que 5 est plus grand que -2 ? true »**

puts "Est-ce que 5 est plus grand que -2 ? #{5 > -2}"

**# Il y aura un test de véracité dans la phrase, ce qui affichera :**

**« Est-ce que 5 est supérieur ou égal à -2 ? true »**

puts "Est-ce que 5 est supérieur ou égal à -2 ? #{5 >= -2}"

**# Il y aura un test de véracité dans la phrase, ce qui affichera :**

**« Est-ce que 5 est inférieur ou égal à -2 ? false »**

puts "Est-ce que 5 est inférieur ou égal à -2 ? #{5 <= -2}"

## Exo 6

exo\_06.rb

**#On attribue une valeur aux 3 variables suivantes**

number\_of\_hours\_worked\_per\_day = 10

number\_of\_days\_worked\_per\_week = 5

number\_of\_weeks\_in\_THP = 11

**#On utilise les variables précédemment défini pour réaliser un calcul (10\*5\*11) qui sera insérer dans la phrase à afficher. Le message affiché sera donc : « Travail : 550 »**

puts "Travail : #{number\_of\_hours\_worked\_per\_day \* number\_of\_days\_worked\_per\_week \* number\_of\_weeks\_in\_THP}"

**#on refait comme précédemment un calcul qui sera insérer dans une phrase à afficher. Mais cette fois ci nous aurons un message d’erreur car la variable « number\_of\_minutes\_in\_an\_hour » n’a pas été défini précédemment. Pour résoudre le beug il nous suffit donc d’attribuer une valeurs de 60 à cette variable.**

puts "Et en minutes ça fait : #{number\_of\_minutes\_in\_an\_hour \* number\_of\_hours\_worked\_per\_day \* number\_of\_days\_worked\_per\_week \* number\_of\_weeks\_in\_THP}"

## Exo 7

### exo\_07\_a.rb

« gets.chomp » permet de faire taper une information par le client. Dans le cas présent son prénom.

puts "Bonjour, c'est quoi ton blase ?"

user\_name = gets.chomp

puts user\_name

### exo\_07\_b.rb

puts "Bonjour, c'est quoi ton blase ?"

print "> "

user\_name = gets.chomp

puts user\_name

### exo\_07\_c.rb

user\_name = gets.chomp

puts user\_name

## Différence entre les trois versions. :

Le premier pose la question avent de demander l’information à l’utilisateur avent d’afficher son prénom.

Le deuxième cas est presque identique mais affichera juste en plus le texte « > » sans saut à la ligne puisqu’il utilise « print ».

Le troisième cas quand a lui ne pose pas la question à l’utilisateur et le met directement face à une information à rentrer sans lui spécifier ce qui est attendu. Avant de lui afficher celle-ci après.

## Exo 8

exo\_08.rb

puts "Bonjour, quel est ton prénom ?"

user\_name = gets.chomp

puts « bonjour, #{user\_name} ! »

## Exo 9

exo\_09.rb

puts "Bonjour, quel est ton prénom ?"

user\_name = gets.chomp

puts "Quel est ton nom de famille ?"

family\_name = gets.chomp

puts « bonjour, #{user\_name} #{family\_name} ! »

## Exo 10

exo\_10.rb

puts « Bonjour, quel est ton année de naissance »

bird year = gets.chomp

puts « En 2017 tu avais #{2017 - bird year} ans »

## Exo 11

exo\_11.rb

puts « Bonjour, peux-tu me donner un nombre ? »

number\_loop = gets.chomp

number\_loop.times do

puts « Salut, ça farte ? »

end

## Exo 12

exo\_12.rb

puts « Bonjour, peux-tu me donner un nombre ? »

number\_loop = gets.chomp

number\_loop.times do i

puts i+1

end

## Exo 13

exo\_13.rb

today\_years = 2020

puts « quel est ton année de naissance ? »

bird\_years = gets.chomp

number\_years = today\_years - bird\_years + 1

number\_years.times do i

puts i + bird\_years

end

## Exo 14

exo\_14.rb

puts « peux-tu donner un nombre ? »

number = gets.chomp

number\_loop = number + 1

number\_loop.times do i

puts number - i

end

## Exo 15

exo\_15.rb

today\_years = 2020

puts « quel est ton année de naissance ? »

bird\_years = gets.chomp

number\_years = today\_years - bird\_years + 1

number\_years.times do i

puts « tu avais #{i} ans en #{i + bird\_years}

end

## Exo 16

Puts « quel est ton âge ? »

age = gets.chomp

age.times do i

puts « il y a #{age – 1 – i} ans tu avais #{ i + 1} ans

end

## Exo 17

Puts « quel est ton âge ? »

age = gets.chomp

age.times do i

if (age – 1 – i ) ==( i + 1 )

puts « Il y a #{age – 1 – i} ans, tu avais la moitié de l'âge que tu as aujourd'hui

else

puts « il y a #{age – 1 – i} ans tu avais #{ i + 1} ans

end

## Exo 18

exo\_18.rb

50.times do i

if i+1 < 10

puts « jean.dupont.0#{i+1}@email.fr »

else

puts « jean.dupont.#{i+1}@email.fr »

end

end

## Exo 19

exo\_19.rb

i= 1

25.times do i

if i\*2 < 10

puts « jean.dupont.0#{i\*2}@email.fr »

else

puts « jean.dupont.#{i\*2}@email.fr »

end

end

## Exo 20

exo\_20.rb

line= « # »

puts « Salut, bienvenue dans ma super pyramide ! Combien d'étages veux-tu ? »

level = gets.chomp

puts « > #{level} »

puts « Voici la pyramide : »

level.times do

if line == #

puts line

else

line=line #{#}

puts « #{line}

end

end

## Exo 21

pyramide.rb

line= « # »

gap= « »

puts « Salut, bienvenue dans ma super pyramide ! Combien d'étages veux-tu ? »

level = gets.chomp

puts « > #{level} »

puts « Voici la pyramide : »

level.times do i

def\_line= line

number= level – (i+1)

if line == #

number.times do

def\_line=gap #{def\_line}

end

puts def\_line

line = line #{#}

else

number.times do

def\_line=gap #{def\_line}

end

puts def\_line

line = line #{#}

end

end

## La pyramide, avec une boucle de champions

line= « # »

n = 1

puts « Salut, bienvenue dans ma super pyramide ! Combien d'étages veux-tu ? »

level = gets.chomp

puts « > #{level} »

puts « Voici la pyramide : »

while n <= level

if line == #

puts line

else

line=line #{#}

puts « #{line}

end

end

## Pyramide de champions, avec une boucle de champions

line= « # »

gap= « »

n = 1

puts « Salut, bienvenue dans ma super pyramide ! Combien d'étages veux-tu ? »

level = gets.chomp

puts « > #{level} »

puts « Voici la pyramide : »

while n <= level

def\_line= line

number= level – n

if line == #

number.times do

def\_line=gap #{def\_line}

end

puts def\_line

line = line #{#}

else

number.times do

def\_line=gap #{def\_line}

end

puts def\_line

line = line #{#}

end

end