Ruby 1/2 - TD

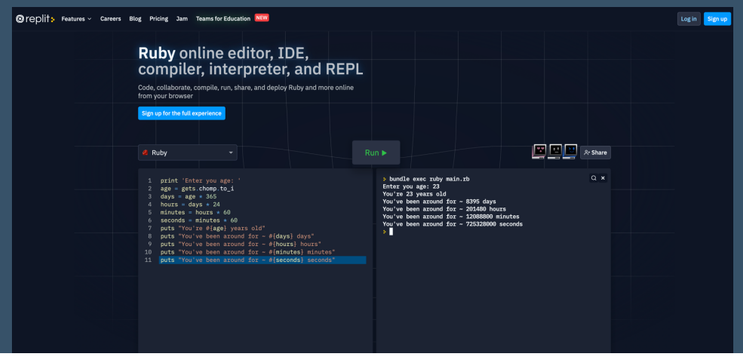
- Ruby recap
- you can use stack overflow / google, use it
- launch the VM
- launch Sublime Editor
- launch a terminal
  - using `irb`

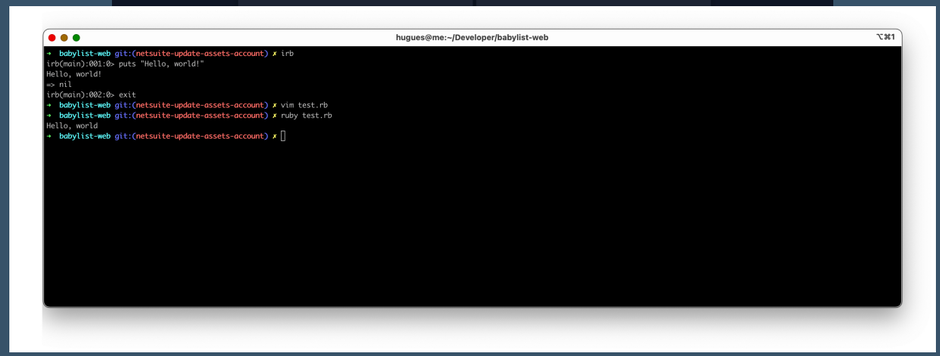
```
➜  hugues✗ irb
irb(main):001:0> puts "Hello, world!"
Hello, world!
=> nil
irb(main):002:0> exit
➜  hugues ✗
```

  - using a file

```
➜  hugues ✗ vim td.rb
➜  hugues ✗ ruby td.rb
Hello, world
```

- alternatively, use an online ruby interpreter -> https://replit.com/languages/ruby





User age

Display some information about the user age

- `gets` **Kernel** method (ie.: available from anywhere) prompt user for input and return a `String`
  - https://www.rubydoc.info/stdlib/core/Kernel:gets
- `chomp` is a **String** method which remove any extra white space
  - https://www.rubydoc.info/stdlib/core/String#chomp-instance_method
- `to_i` is a **String** (and other class) method, which attempt to convert the string to an integer (or return 0)
  - https://www.rubydoc.info/stdlib/core/String#to*i-instance*method

```
Enter you age: 21
You're 21 years old
You've been around for ~ 7665 days
You've been around for ~ 183960 hours
You've been around for ~ 11037600 minutes
You've been around for ~ 662256000 seconds
```

```ruby
print 'Enter you age: '
age = gets.chomp.to_i
days = age * 365
hours = days * 24
minutes = hours * 60
seconds = minutes * 60
puts "You're #{age} years old"
puts "You've been around for ~ #{days} days"
puts "You've been around for ~ #{hours} hours"
puts "You've been around for ~ #{minutes} minutes"
puts "You've been around for ~ #{seconds} seconds"
```

Guess the number

The user has 6 attempts to guess a (random) number between 0 and 100, on each attempt tell the user is the number is greater or lower

- `while` loop or `for in` loop
- `rand(max)` is a **Kernel** method which return a number between 0 and max
  - https://www.rubydoc.info/stdlib/core/Kernel#rand-instance_method

```
Enter a number between 0 and 100? 50
too low, 5 attempts left
Enter a number between 0 and 100? 75
too big, 4 attempts left
Enter a number between 0 and 100? 62
too big, 3 attempts left
Enter a number between 0 and 100? 55
too low, 2 attempts left
Enter a number between 0 and 100? 59
too low, 1 attempts left
Enter a number between 0 and 100? 61
you lost, the number was 60
```

```ruby
i = 5
number = rand(100)
while i >= 0
  print 'Enter a number between 0 and 100? '
  guess = gets.chomp.to_i
  case
  when number == guess
    puts 'you won!'
    break
  when guess > number
    puts "too big, #{i} attempts left"
  when guess < number
    puts "too low, #{i} attempts left"
  end
  i -= 1
end
puts "you lost, the number was #{number}" if i <= 0
```

Guess the user age

Given the following function (note the :)

```ruby
require 'json'
require 'net/http'

def age(firstname)
  raise 'Missing name' unless firstname
  url = "https://api.agify.io/?name=#{firstname}"
  response = {
          "name" => "hugues",
          "age"=>  49,
          "count"=>  4421
  }
  pp response
  response['age']
end
```

```
What's your name?
hugues
You're probably ~49 years old
```

Prompt for user first name and display his age

```
puts "What's your name?"
name = gets.chomp()
age = age(firstname: name)
puts "You're probably ~#{age} years old"
```

Slack API parsing

Given the following method

```ruby
require 'uri'
require 'net/http'

def get_conversations
  uri = URI('https://slack.com/api/conversations.list?limit=50')
  req = Net::HTTP::Get.new(uri)
  req['Authorization'] = "Bearer xoxp-2486113197334-2492860403907-2492926538098-76ac2d6b0dcc5d6a24b3c72889355468"
  res = Net::HTTP.start(uri.hostname, uri.port, use_ssl: true) { |http| http.request(req) }
  response = res.body
  JSON.parse(response)['channels']
end
```

```ruby
conversations = get_conversations
puts conversations.inspect
# => [{}, {}, ...]
puts "first conversation id: #{conversations[0]['id']}"
# => first conversation id: C02E24403SB
```

```
{
    "ok": true,
    "channels": [
        {
            "id": "C02E244033R",
            "name": "general",
            "is_channel": true,
            "is_group": false,
            "is_im": false,
            "created": 1631693335,
            "is_archived": false,
            "is_general": true,
            "unlinked": 0,
            "name_normalized": "general",
            "is_shared": false,
            "parent_conversation": null,
            "creator": "U02EGRABVSP",
            "is_ext_shared": false,
            "is_org_shared": false,
            "shared_team_ids": [
                "T02EA3BST9U"
            ],
            "pending_shared": [],
            "pending_connected_team_ids": [],
            "is_pending_ext_shared": false,
            "is_member": true,
            "is_private": false,
            "is_mpim": false,
            "topic": {
                "value": "",
                "creator": "",
                "last_set": 0
            },
            "purpose": {
                "value": "This is the one channel that will always include everyone. It's a great spot for announcements and team-wide conversations.",
                "creator": "U02EGRABVSP",
                "last_set": 1631693335
            },
            "previous_names": [],
            "num_members": 1
        },
        {
            "id": "C02EGRBMQPX",
            "name": "http",
            "is_channel": true,
            "is_group": false,
            "is_im": false,
            "created": 1631693553,
            "is_archived": false,
            "is_general": false,
            "unlinked": 0,
            "name_normalized": "http",
            "is_shared": false,
            "parent_conversation": null,
            "creator": "U02EGRABVSP",
            "is_ext_shared": false,
            "is_org_shared": false,
            "shared_team_ids": [
                "T02EA3BST9U"
            ],
            "pending_shared": [],
            "pending_connected_team_ids": [],
            "is_pending_ext_shared": false,
            "is_member": true,
            "is_private": false,
            "is_mpim": false,
            "topic": {
                "value": "",
                "creator": "",
                "last_set": 0
            },
            "purpose": {
                "value": "This #channel+ is for working on a project. Hold meetings, share docs, and make decisions together with your team.",
                "creator": "U02EGRABVSP",
                "last_set": 1631693553
            },
            "previous_names": [],
            "num_members": 1
        },
        {
            "id": "C02EGRBFFMQ",
            "name": "random",
            "is_channel": true,
            "is_group": false,
            "is_im": false,
            "created": 1631693335,
            "is_archived": false,
            "is_general": false,
            "unlinked": 0,
            "name_normalized": "random",
            "is_shared": false,
            "parent_conversation": null,
            "creator": "U02EGRABVSP",
            "is_ext_shared": false,
            "is_org_shared": false,
            "shared_team_ids": [
                "T02EA3BST9U"
            ],
            "pending_shared": [],
            "pending_connected_team_ids": [],
            "is_pending_ext_shared": false,
            "is_member": true,
            "is_private": false,
            "is_mpim": false,
            "topic": {
                "value": "",
                "creator": "",
                "last_set": 0
            },
            "purpose": {
                "value": "This channel is for... well, everything else. It's a place for team jokes, spur-of-the-moment ideas, and funny GIFs. Go wild!",
                "creator": "U02EGRABVSP",
                "last_set": 1631693335
            },
            "previous_names": [],
            "num_members": 1
        },
        {
            "id": "C02ETGJE76F",
            "name": "test",
            "is_channel": true,
            "is_group": false,
            "is_im": false,
            "created": 1631880599,
            "is_archived": false,
            "is_general": false,
            "unlinked": 0,
            "name_normalized": "test",
            "is_shared": false,
            "parent_conversation": null,
            "creator": "U02EGRABVSP",
            "is_ext_shared": false,
            "is_org_shared": false,
            "shared_team_ids": [
                "T02EA3BST9U"
            ],
            "pending_shared": [],
            "pending_connected_team_ids": [],
            "is_pending_ext_shared": false,
            "is_member": true,
            "is_private": false,
            "is_mpim": false,
            "topic": {
                "value": "",
                "creator": "",
                "last_set": 0
            },
            "purpose": {
                "value": "",
                "creator": "",
                "last_set": 0
            },
            "previous_names": [],
            "num_members": 1
        }
    ],
    "response_metadata": {
        "next_cursor": ""
    }
}
```

Return the number of conversations

— https://www.rubydoc.info/stdlib/core/Array#count-instance_method

```
4 conversations
```

```
conversations = get_conversations
puts "#{conversations.count} conversations"
# => 4 conversations
```

Display the name of each conversations and it's creation date

```
general created at 2021-09-15 10:08:55 +0200
http created at 2021-09-15 10:12:33 +0200
random created at 2021-09-15 10:08:55 +0200
test created at 2021-09-17 14:09:59 +0200
```

- `for each`
- `Time.at(epoc)` is a method which take an epoc (seconds since Jan 1 1970) -- (`created` is an epoc)
  - https://www.rubydoc.info/stdlib/core/Time.at

```ruby
conversations = get_conversations
for conversation in conversations
  puts "#{conversation['name']} created at #{Time.at(conversation['created'])}"
end
```

Define a method to grab the list of messages for a conversations
To grab a list of messages "https://slack.com/api/conversations.history?limit=50&channel=CHANNEL_ID"

```
def get_messages(channel_id)
  # do something
```

```
get_messages('C02E24403SB')
# => [{}, {}, ...]
```

```ruby
require 'uri'
require 'net/http'

def get_messages(channel_id)
  uri = URI("https://slack.com/api/conversations.history?limit=50&channel=#{channel_id}")
  req = Net::HTTP::Get.new(uri)
  req['Authorization'] = "Bearer xoxp-2486113197334-2492860403907-2492926538098-76ac2d6b0dcc5d6a24b3c72889355468"
  res = Net::HTTP.start(uri.hostname, uri.port, use_ssl: true) {|http| http.request(req) }
  response = res.body
  JSON.parse(response)['messages']
end
```

For each channel, get the messages and display the first 3

— for in

```
fetching messages of C02E24403SB
Hi
 desoler gael first
 :wave: coucou grp2

fetching messages of C02EDRBMZPX
:wave: Hi everyone!
dsdf
dfsfds

fetching messages of C02EGRBFMRQ
:wave: lllllllllllllllll
 Hi !
:rire:okkkkkk

fetching messages of C02ETGJE76F
:wave: Hi everyone!
:wave: Hi everyone!
:wave: Hi everyone!
```

```ruby
for conversation in conversations
  conversation_id = conversation['id']
  puts "fetching messages of #{conversation_id}"
  messages = get_messages(conversation_id)
  count = 0
  for message in messages
    puts message['text']
    count += 1
    break if count >= 3
  end
  puts ""
end
```

Run a loop which ask for a conversation id and display all messages, break if user enter `exit`

— `loop do` or `while true`

— `break`

```
conversation id? C02E24403SB
Hi
 desoler gael first
 :wave: coucou grp2
:wave: hello!
:wave: Hi everyone!
:wave: Hi everyone!
aaa
...
<@U02EGRABVSP> has joined the channel
conversation id? exit
=> nil
```

```ruby
while true
  print 'conversation id? '
  conversation_id = gets.chomp
  break if conversation_id == 'exit'

  messages = get_messages(conversation_id)
  puts messages.map { |m| m['text'] }.join("\n")
end
```

Make this loop return inform if the conversation id is invalid

- `get_messages` should return `nil` if the conversation is invalid

```
conversation id? 123
this conversation does not exist?
conversation id? exit
=> nil
```

```ruby
while true
  print 'conversation id? '
  conversation_id = gets.chomp
  break if conversation_id == 'exit'

  messages = get_messages(conversation_id)
  if messages
    puts messages.map { |m| m['text'] }.join("\n")
  else
    puts "this conversation does not exist?"
  end
end
```