Main

```
- Create a dataframe to record the number of refs collected on each page
    - Structure: [page, mix_num_refs, template_num_refs, text_num_refs,
      total_num_refs]
    - Name: df_ref_number
- Create a dataframe to load the references
    - Structure: [page, ref_type, ref_extracted, template,
      ref_no_template, parsed_template]
    - Name: df_refs
- Create a list of <*.JSON> files in the wiki_pages directory
    - Name: json_files_wiki_pages_directory
- Loop over the N elements of json_files_wiki_pages_directory:
    - Open the file "<PATH>" + "<FILE_NAME>" + ".json"
        - Read the "source" key (to get text)
            - Output: source_text
    - Extract references:
        - Input: source_text
        - Output: refs
    - Add refs to df_refs
        - Input: refs
    - Remove duplicated items from df_refs based on "template"
    - Parse templates
        - Input: df_refs
        - Output: df_refs' new columns
    - Add the number of extracted references to df_ref_number
        - Input: df_refs
        - Output: df_ref_number
- Save df_refs
    - Output: wikipedia_references.csv
- Save df_ref_number
    - Output: wikipedia_number_of_references.csv
```

Extract references:

Input: source_text

```
- Extract refs from source only between tags
    - Input: source_text
    - Output: ref_list_tags,
              ~~number_ref_tags~~,
              page_off_tags
- Extract refs from page off tags source only template
    - Input: page_off_tags
    - Output: ref_list_off_tags_only_templates,
                    ~~Number_refs_off_tags_only_templates~~,
                    page_off_tags_templates
  - Extract refs from page off tags and templates (references sections)
      - Input:  page_off_tags_templates
      - Output: ref_list_only_ref_sections,
                    ~~number_only_ref_sections~~
  - Join ref lists
      - Input: ref_list_tags,
                    ref_list_off_tags_only_templates,
                    Ref_list_only_ref_sections
      - Output: refs
Output: refs
```

Extract refs from source only between tags

```
Input: source_text
  - Get text between tags ”<ref...<ref/>”
      - Input: source text
      - Output: between_tag_list (List of text between tags)
  - Remove tags + text between tags from the source text
      - Input: source text
      - Output: page_off_tags
  - Create a list to save extracted references
      - Output: ref_list
  - Loop over the M elements of the between_tag_list:
      - Extract template(s):
          - Input: $text_i$
          - Output: template
      - if templates:
          - text_no_template = remove($text_i$, pattern=template)
          - if len(text_no_template) > len($text_i$)*0.2:
              - ref_list.append((‘mix’, $text_i$, templates,
                text_no_template))
          - else:
              - ref_list.append((‘template’,text, templates, None))
      - else:
          - ref_list.append((‘text’, text, None, None))
Output: ref_list, len(ref_list), page_off_tags,
```

Extract refs from cleaned source only template

```
Input: page_off_tags
    - wikicode = mwparserfromhell.parse(page_off_tags)
    - templates = wikicode.filter_templates()
    - Page_off_tags_templates= remove(text_i, pattern=templates)
    - Ref_list = []
    - For temp in templates:
          - Ref_list.append((‘template’, temp, temp, None))
Output: Ref_list, len(ref_list), Page_off_tags_templates
```

Extract refs from page_off_templates (**references sections)**

```
Input:  page_off_tags_templates
    - end_text_section= [“== Reference”, “==Reference”, “== reference”,
      “==reference”, “== Citatio”, “==Citation”, “==citation”, “==citation”,
      “== Source”, “==Source”, “==source”, “==source”, “== Bibliography”,
      “==Bibliography”, “==bibliography, “==bibliography”, “== Note”,
      “==Note”, “==note”, “==note”]
    - text_ref_sections = page_off_templates.split(end_text_section,
      obs=first)
    - refs_asterisco = find.all(text_ref_sections, pattern=’* letter or
      *letter’)
    - Ref_list =[]
    - For ref in ref_arter_asterisco:
          - Ref_list.append((‘text’, ref, None, None))
Output: Ref_list, len(Ref_list)
```