UI task:

* Based on phenomenon that from almost every article on English Wikipedia it is possible to get to article about „Philosophy“ (<https://en.wikipedia.org/wiki/Philosophy>) by clicking first link.
* Opening random article on Wikipedia:
  + Application selects random article from Wikipedia main page „On this day“ section.
  + In „On this day“ section there shall be different articles everyday.
* Clicking first link in the article until getting to article „Philosophy“:
  + First link shall meet following criteria:
    - It is the first link in main article text. Not the link in info boxes, or info lines right below article title as clicking such a link will often lead to a loop. For example at article „Donner Party“ (<https://en.wikipedia.org/wiki/Donner_Party>), clicking very first link „[Donner Party (disambiguation)](https://en.wikipedia.org/wiki/Donner_Party_(disambiguation))“ leads to a page „Donner Party (disambiguation)“ (<https://en.wikipedia.org/wiki/Donner_Party_(disambiguation)>) where the first link leads back to article „Donner Party“. So at this „Donner Party“ article the first valid link is „[American pioneers](https://en.wikipedia.org/wiki/American_pioneer)“.
    - It is not external link, so it does not lead outside of Wikipedia.
    - It is not red link, so it does not lead to non-existent pages. An article with first red link is „Thout 1“ (<https://en.wikipedia.org/wiki/Thout_1>).
    - It points to article on English Wikipedia, so it is not a small link in parentheses so a citation or a link to other language Wikipedia mutation. Such a first link is at article „National Democratic Institute“ (<https://en.wikipedia.org/wiki/National_Democratic_Institute>).
* To printing out number of redirects, there is also added a list of all visited articles, their title and URL as this information is collected for loop checking.
* Possible corner cases that are handled:
  + Very first random article opened from Wikipedia main page „On this day“ section is „Philosophy“ so no first link clicking and successfully stopping application as „Philosophy“ article was reached with 0 redirects with displayed list of all visited articles, in this case there is only „Philosophy“ article.
  + Loop:
    - A loop may occur so during clicking first link an already visited article is opened. The application checks whether or not opened article was already visited by storing titles and URLs of all already visited articles and checking if title and URL of actually opened article is present among these already visited articles. Application checks both title and URL as I don’t know Wikipedia well but I noticed there may be something like title prefix so maybe this could have an impact on comparing only titles of articles. And comparing only by URL is not so descriptive and not clear which article it is, so I decided to check both. Once loop occurs, program is stopped with message informing on loop and article that looped and with displayed list of all visited articles.
    - Example of loops:
      * A loop occurs when going from article „German submarine U-505“ (<https://en.wikipedia.org/wiki/German_submarine_U-505>), it occurs for article „German language“ (<https://en.wikipedia.org/wiki/German_language>).
      * A loop may also occur when going from article „Allies of World War II“ (<https://en.wikipedia.org/wiki/Allies_of_World_War_II>) for article „Awareness“ (<https://en.wikipedia.org/wiki/Awareness>). It seems there are being changes made at „Awareness“ article that sometimes remove the loop, sometimes introduce it back.
  + As I understood there are some articles from which it is not possible to get to „Philosophy“ so probably it might happen that application would be clicking endlessly through Wikipedia. Therefore there is a limit for number of redirects, it is set to by default 50 as it seems 50 is enough for vast majority of cases. Can be changed in file „ui\_task\_test\_data.py“ by changing value of „max\_redirects\_limit“ variable. If the limit is reached the execution is stopped with message saying that maximum number of redirects was performed with displayed list of all visited articles.
  + Articles without any links:
    - There should be dead end pages but there is none in this category (<https://en.wikipedia.org/wiki/Special:RandomInCategory/All_dead-end_pages>).
    - Reportedly it was article „Karoline Hausted“ (<https://en.wikipedia.org/wiki/Karoline_Hausted>) but it is fixed now and there are links allowing to get to „Philosophy“ article.
    - Seems the only pages without any links are non-existent articles like „Nesi 6“ (<https://en.wikipedia.org/wiki/Nesi_6>).
    - If application comes to such an article the execution is stopped with message saying that it is not possible to continue from that article probably because of absence of valid link. List of all visited articles is displayed as well as stack trace.
  + Article that is very far away from „Philosophy“ article. I found information that such an article is „Parmouti 12“ (<https://en.wikipedia.org/wiki/Parmouti_12>) with around 240 redirects. Based on that I found article that is even more redirects away from „Philosophy“, it is „Paoni 9“ (<https://en.wikipedia.org/wiki/Paoni_9>) with almost 300 redirects. If application comes to such an article it clicks through the links up to „Philosophy“ article. In order to do so, limit for number of redirects has to be increased.

API task:

* „Using the described REST API, select and write to console 5 photographs taken in 2018,

where there is the surface of the planet Mars visible or were taken from the Mars surface“:

* + Seems that easiest way of getting 5 photographs showing surface of Mars or taken from its surface is simply to take first 5 photographs returned by the request, i.e. find a request which returns at least first 5 photographs meeting this criterion:
    - I tried requests with different parameters (q, description, description\_508, keywords, location, title) and values (mars, mars surface, taken from mars surface):
      * In majority of cases among the first 5 photographs returned there was at least one photograph not showing Mars surface or not taken from its surface, like artist’s rendition (<https://images-assets.nasa.gov/image/PIA22228/PIA22228~thumb.jpg>) or there were no results returned at all.
      * Successful was setting „q=taken from mars surface“ as well as „description=taken from mars surface“ (both returned the same results) and also „keywords=mars“. I chose „keywords=mars“ as it seems to be the most reliable as first 5 pohotographs show surface of Mars and also other photographs returned that I tried opening showed Mars surface.
  + 5 photographs are handled directly in request via parameter „page\_size=5“ (in application the number can be set via variable „number\_of\_photos“ in „api\_task\_test\_data.py“ file), so that request returns only 5 photographs as it is bit faster and requires less server resources than returning more results, moreover it is useless to return more results that are not needed and also it makes extraction of data from response easier as application can extract data from all returned photographs.
  + Taken in 2018, I assume this means „YEAR PUBLISHED“ at page <https://images.nasa.gov> and refers to request parameters „year\_start“ and „year\_end“ (in application the year can be set via variables „year\_start\_photos“ and „year\_end\_photos“ in „api\_task\_test\_data.py“ file) – correct assumption as confirmed by RWS. This should also lead to consistency in returned results as no new photographs can be added in year 2018 so the same results shall be returned for the same request call. Though there can be a confusion because it seems this does not necessarily mean year when photograph was taken but year when photograph was published. Because when searching for images at page <https://images.nasa.gov> with query „taken from mars surface“ and „YEAR PUBLISHED“ as „2018“ or via request „https://images-api.nasa.gov/search?q=taken from mars surface&media\_type=image&year\_start=2018&year\_end=2018“ then in the results there is also photograph „InSight's Final Location on Mars“ (<https://images.nasa.gov/details/PIA22878>) and in its description there is stated that the photograph was taken in 2015.
  + It is not stated which information shall be written to console regarding 5 photographs as it is up to me. For each photograph I decided to write its title, creation date, description and link to photograph preview.
  + There are assertions added, there is assertion for 200 status code, number of obtained photos, for each photo a creation year is asserted.
* „Using the described REST API, for keyword Mars taken in 2018 with media\_type="video"

write out every link for video file for the first 5 results. Hint: search results includes a

collection of links, which can contain a link to video file type“:

* + I used parameter „keywords=Mars“ with capital „M“ as stated in description even though providing mars returns the same results, so it seems providing „Mars“ or „mars“ have no effect on request.
  + First 5 results are handled in the same way as for photographs, so directly in request via parameter „page\_size=5“ (in application the number can be set via variable „number\_of\_videos“ in „api\_task\_test\_data.py“ file).
  + Taken in 2018 is handled in the same way as for photograhps so via request parameters „year\_start“ and „year\_end“ (in application the year can be set via variables „year\_start\_videos“ and „year\_end\_videos“ in „api\_task\_test\_data.py“ file).
  + As far as I understood getting every link to video file can be done in 2 ways:
    - Via prepared request for retrieving a media asset’s manifest, so „GET /asset/{nasa\_id}“.
    - By calling a get request to address with „collection.json“ file that is returned in response as „href“ for each video file.
    - I chose to call a get request to address with „collection.json“ file as that is mentioned in task description and it is also easier to extract video files from there as the response is just a list of all media files related to the particular video file while „GET /asset/{nasa\_id}“ returns more information and the list of all media files related to the particular video is stored inside „items“ that is inside „collection“.
    - Video files are recognized via extension „mp4“ (can be set in application via variable „video\_file\_extension“ in „api\_task\_test\_data.py“ file).
  + There are assertions added, there is assertion for 200 status code and number of obtained videos. There is no assertion for creation year as creation date is not written into console.