The Hieroglyphics Initiative takes an image of hieroglyphic text and through a series of steps, suggests a translation.

Ḥtp ḏi nsw.t Wsỉr… → That which the king Osiris gives…

Images provided by Alex Fry, Technology Director, Psycle
Property of Ubisoft Hieroglyphics Initiative Workbench

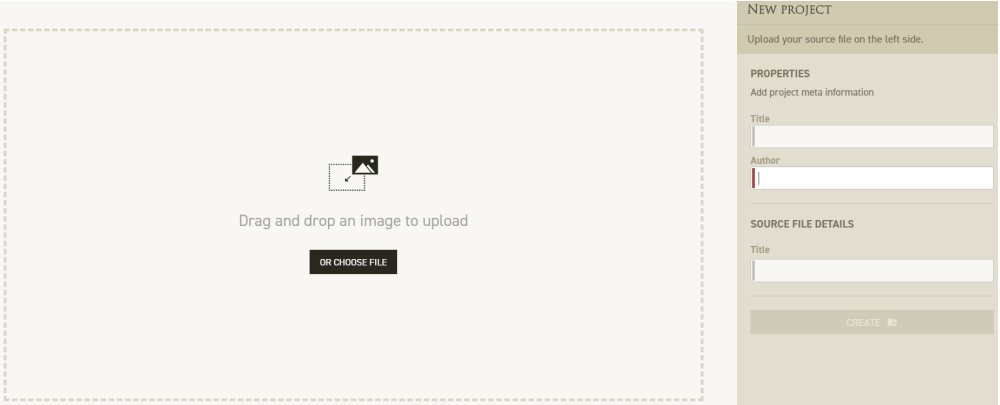
I developed a script that can extract the image files in someone else's project so you can use them.

Transfer YML file over internet → **Run R Script** → **Extract source PNG image**


```
library(yaml)
library(base64enc)
library(magick)
yamldata <- yaml.load_file(file.path("data","extract-so
#image <- yamldata[['files']]
for (source in yamldata$files$sources)
{
  imageData <- (source$dataURL)
}
imagesource <- gsub("^data.*base64, *", "", imageData)
decodedimagesource <- base64decode(imagesource)
writeBin(decodedimagesource, "source-image.png")
```

Images provided by Alex Fry of Psycle
Property of Ubisoft Hieroglyphics Initiative Workbench

The Hieroglyphics Initiative Workbench



Images provided by Alex Fry of Psycle
Property of Ubisoft Hieroglyphics Initiative Workbench

Auto-Classify Tool: Can the program correctly identify the hieroglyphs?

Images provided by Alex Fry of Psycle
Property of Ubisoft Hieroglyphics Initiative Workbench

How do I turn THIS into a PNG file?

extract-source-images.yml

```
389 files:
390   sources:
391     - id: a7ebf99b-92e0-42ae-8a5e-de2dcda7d53b
392       title: wall_photo
393       dataURL: >-
394         
            M8PMBNH80jFAEzRg2thuAtsY8LgzNn5Ks1VZJZXqXZV2+c77fki8vuusm7fKVXqVLFWp9onIyHx7z2Pvtfcs557w/q7vypnZ2NN
            LVAskCyQLJAsJAAu5pIiQLJAskCyQLJAskCyQLJAskCyQLJAskCyWFFkjA/S4YpNTEZIFkgWSBZIFkgWSBZIFkgWSBBNzTHEgWSBZIFkgWS
            OZAskCyQLJAskCyQLJAskCyQLHAXWCAB97tgkFITkwWSBZIFkgWSBZIFkgWSBZIFEnBPcyBZIFkgWSBZIFkgWSBZIFkgWEAusEA
            CyQLJAssBVIEE3O+CQOpNTBZIFkgWSBZIFkgWSBZIFkgWSMA9zYFkgWSBZIFkgWSBZIFkgWSBZIFkgWSBZIFkgWEAjUh8Fg5SamCyQLJAsk
            BiklMVngzzdA3sxGZha/7/wWpxYmCyQLJAskCyQLL3G0WSMD9hbuxlN5XbIF83mw89h82/s9bwcbjsY3GgM5v33KWs7GNrZAv2HA
            bbB+4vfl4/x/W9mI7vzcfj9nqbjk4WSBZIFkgWSBZIFkgWSBa4uy2QgPvdPX6p9S9igWIxb4OBgOCAei6Xs9FoLPCeu8nsByizT/zk8xw/s
            BZIFkgWSBe48CyTgfueHSWrRqZgBgPdotA8C+28IOz9s0yBdEWCi7lP7TzelUChof/YZDodWKpWs3++/iq197U9lI3Drj4A7s8H5
            B9APnjqqyQ80QH0AesH4jkL/xslfbpLQvHIrp36/WdW4fuL+8yHgC7q/WyKXzJAskCyQLJAvcixZIwPleHPV7rH+A34iU3xh9j
            48bf2fbfjmXSsckCyQLJAskCyQLJAtdbIAH3NCPe8BYAaLPdLDp+K2Acnlerlq1q5xwAeAA73/N3v399FP7lG/PlRazj/UN23yB
            cRbAHwgtk447QXSyWbnZ2ll2UV/czMzNjy8rJ9/etft6eeesoWFxetXC7blStXXJoGz3odkPeLxZx/3bjhlqOvrndywdwz93Qj
            wsD9xuTUerlURNGhyHQ6Lfv+H3iPHThwwE6ePGmmTp353/E9wL5Sqdjm5qatra3Z0tK5WPy5c+fsql/9qn3xil+0p5/8lm0Mzot
            LAPWaBl/zevcfskrr7GlkgeOU3Krk02e7lehPlFv4HSLNf0FVulkQi6lBjoMUQVY+/2ffIkSP28z//8/be975X4BwAp8/b+WyX
            vf3VX/2VNZvNyTnoU3hIctipJRuuS7u63e5+PzJQC6Mn8l+n/y4WzUSnz7ZqtWidDnKV/kf5xwLARKOexYARlv68DYejyefsHOc
            WZWSqWrFqt2+7erog8jFeMRy5ftPHIcwOKpYINBl2tGkxWEV5iBH66D6/RVE2XSRZIFkgWSBZIFnjdLZCA++s+BKkBYYPpBRXXW
            CZzv7elZqTWn8/X6Ht2+tbuNGlvt6X/clawQjFnhXzJDh5asOqSZptb6lYqVuzkqRP6feHCBR0PrabX7dqff+Qj9vGPfcw2t3b
            C9593vsKKlbFsbW2Y7PzvlYtnWltfs4vmL9rl/+JW59rXHrN0dWI5CUlPR9unUqSXnzkBU8Wm3KahZT8V7286TyGfEyXoOKFDt
            bJjx7zEdRasRmYcx0dFPlHpOMqqtTll/EpifAvAuBOQynGlLFkgWSBZIFkgWuJctkID7vTz6d0DfI8kygPrNkkUDuO8nhPYFHG+
            Vm+Xz0ms2WdbsdFWnid6lUtgMHVmxpadl6va6+rSdrNjs7p+vXajXx3ivFKnX6Pdta37B/+sqX7Utf+KL+P3HsuFUbdXvw5Ck7+
            a33LNrY37KE3PW5j3MguX7hszzz3jOlt7smlq5e0CzEKwdWrV7UC0OsOZBscmN3dllUqJUXYWT2YjswXilSCHdlwOLal5Tk7fuy
            9n/6t9/gy/aKe/9YxdXVulbq9vlWpN+/QHrnU/zhR4cHY8x6BnSYrr7BPRvwEXZxh9X7lzH9O7gA+KlIvTje6Aa2GkyQLJAsk
            QWD90UcftUceecQefPBBO3jwoMAsQBGwDl0DvnpIQy4tL9iw37ftnUlFlomvN/fa4mAXC2VrtnZtOBjbcNS3+blFRd72rz/oWy5
            mychn+/cCuXFnVisCXv/xV293dtocftSOHz9q+Tw0mZaAa7vdlMoBEW4cgXMXLli5UreHHnnYFmbn7MrqNXvu9GnbaTXs8oVL5
            7eG3Pmpvfsu0DygFoNnetUgxYbaZhm9c2bXZ+UY7LOJ+zySev2OXLl+3JJ5+0586csU9+8tO2ur6RrZyUtbIR1V5F7blh4kzoPjc
            gHwAbkAopDzpFIOZrqXr2OYO/9wfeICvOud7lLNS8+BSASZd7a2lJEd3d3VyAaoBuUHED0OuKcXbl4QcBSOOxbp9MT5715qQk4N
            BeWD4+jn2tpVWlkSKO74tWuA3JGoJ9/8ShP2+OPfsBMnjtv99S+y+flZKxbLtrAwZltbOwLwnPfSc2dtNBjawtKi7W7v2ZWzV2l
            tXkfzxeCinol6r2Pb2tpULefX/ne98hxykXr9r2xvblivKbXF+wbr9nrjxM3OzVig4dYh+tbod29zYtj/64/9of/EXH7XzF68oq
```

Using THIS R Script:

```
library(yaml)
library(base64enc)

options(warn=-1)

all_yaml_files <- dir(path="data", pattern=".yml", full.names = TRUE)

for (this_yaml_file in all_yaml_files)
{
  yamldata <- yaml.load_file(this_yaml_file)
  for (source in yamldata$files$sources)
  {
    imageData <- (source$dataURL)
    imagesource <- gsub("^data.*base64, *", "", imageData)
    decodedimagesource <- base64decode(imagesource)
    writeBin(decodedimagesource, file.path("output_data", paste(source$title,".png")))
  }
}
```

What am I comparing?

Gardiner Signs	Gardiner Signs
z11	z11
g38	g43
x1	x1
n33a	n33
d44	d40
s43	m35
o1	o1
t15	p8
n35	n18
aa12	n35
d36	f4
a18	d36

Here you can see the correct matches between the two lists of Gardiner Signs:

Gardiner Signs	Gardiner Signs
z11	z11
g38	g43
x1	x1
n33a	n33
d44	d40
s43	m35
o1	o1
t15	p8
n35	n18
aa12	n35
d36	f4
a18	d36

And here you can see the matches it *almost* got correct...

Gardiner Signs	Gardiner Signs
z11	z11
g38	g43
x1	x1
n33a	n33
d44	d40
s43	m35
o1	o1
t15	p8
n35	n18
aa12	n35
d36	f4
a18	d36

You can view a section of my R script which compares the lists of Gardiner Signs below:

```
DataVector1 <- vector("character", length = 0)

for (sentence in yamldata1$project$sentences)
{
  for (word in sentence$words)
  {
    for (glyph in word$glyphs)
    {
      DataVector1 <- c(DataVector1, glyph$gardinerCode)
    }
  }
}
yamldata2 <- yaml.load_file(file.path("data","correct-gardiner-codes.yml"))
DataVector2 <-vector("character", length = 0)

for (sentence in yamldata2$project$sentences)
{
  for (word in sentence$words)
  {
    for (glyph in word$glyphs)
    {
      DataVector2 <- c(DataVector2, glyph$gardinerCode)
    }
  }
}
```

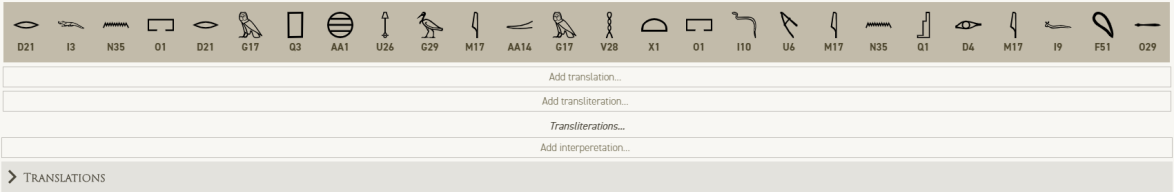
(It's too long to fit on this page!)

Running this script gives output that looks like this:

```
[1] "d36"
[1] "f4"
[1] 11
[1] 3
[1] "a18"
[1] "d36"
[1] 12
[1] 3
> PercentMatch <- MatchCount/NumberOfRecords*100
> print(format(PercentMatch, digits=2, nsmall=2))
[1] "25.00"
```

You can see that the script returned a percentage of **25**.
The auto-classify tool **correctly** identified **3** out of the **12** selected hieroglyphs.

So what's next?



Continue developing codes that test various aspects of the program.
Let's translate some texts and see how we go!
See for yourself!