```
In [1]: import pandas as pd #needed
        from PIL import Image
        import requests
        from io import BytesIO
```

```
In [2]: pic_url = 'https://steamcdn-a.akamaihd.net/steam/apps/730/logo.png' # Successful, but white backgroup.
```

```
In [3]: r = requests.get(pic_url)
```

Source to download image from online

https://stackoverflow.com/questions/7391945/how-do-i-read-image-data-from-a-url-in-python (https://stackoverflow.com/questions/7391945/how-do-i-read-image-data-from-a-url-in-python)

Working with Image

```
In [4]: # storing the picture in Bytes forms
        img = Image.open(BytesIO(r.content))
```

```
In [5]: img
```

```
Out[5]:
```

In [6]:

```
# saving in the pics folderdata:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAk8AAACJCAQAAACf4BRlAAAd+0lEQVR4nO2d
eZgU1bXAfzPAgKzDvimyiewyKIuAyqKoICLKRBFUFBXJwwjocw/BJRqXAJpEjXFBETdEiYggKqCIIqiAgAIPFRJQdBCHYREGmPv+6Kanu/buu
lXdg+fX3zdf1723zr1dU3XqLueek66UQBEHIRLLT3QBBsCCbwbzCVhSKfXzMLdRMd5OE8MmS3pOQcfTgCToY0gq5jX8it+tvCuk9CZnGaD40KS
fI5XHmc0oa2iOkDek9CZnFpcxwzF/AKLaG1BYhzUjvScgkmvKkS4n+DA2lJUIGIOpJyCT+TBXXMtVCaIeQEcjgTsgcGrGF8q6lNnJiCG0RMgD
pPQmZw4UelBO0ok3gLREyAlFPQubQ12O5HoG2QsgYRD0JmcNJHss1CLQVYG6knIHBp6LFch0FYIGYOoJyFTyOEYjyV3B9o0IWMQ9SRkCsXs
91hyfaDtEDIGUU9C5vCBhzK/Mp0F0e91fdY3lLY+JQiBIupJyBzmuOSXcAcduJyDAGSz0Gd9Y+nqU4IQKF7sTAQhKLI5jf6cSE22cBOfuZTez
X1xR/WpH2TThPQj6klIHxfwEC1jR12p5FK+BnUpiB2dSR3+yrvMD6h1QToOpnuKLZid4d4nlCaM
ieOyEdZPEUV6Vw3odUoYnvKfEI/8Ol5DKHf/GtFnmCdkQ9CengBqamuwkxWrMh4bgGB9mXprYICYh6EsLnRL4kJ92NiDGFooTjMWSzF4BJTEt
De4QYMvckhM/kjFBO+6lAOeASihPSa1KBOgA0TkOrhDik9ySETWc+T3cTABjPW1QF1nA4Ib0fe9kOwM+yfSa9SO9JCJuxIdf3ZZi6
+wG2RUgK6T0J4VKRn6geYn2d2dWRlibYJWpPckhEvPQJRTEfuYTTbXkJWQPsOm5ySUCWTPnRAufQKRWp0XGMNo/pOQ+gQjJHBnWUZ6T0K45AUid
RUbyKUmuXFpk7g/kLqE0JC5JyFcNnO8ZonF3Bh9zQ6lZ1x6Lrs01ySEjKgnIUzKs59ymmu8BdjhLNNQkhI3NPQphU066cII/RQF
1mGGS3tC4ulB1EPQlhUiMQqZOoyj9N8Vt6WpYVyhCinoSyTz3eZYgp9dQ0tETQiqgnIUy8BjtIFiufT6cFVJcQGjI1LoRJDvsNhpNB0ozNodU
lBID0noQwKQ5JYRRxI53YEkpdQmCIWaYQLqtpFkItHzE5hFqEgBH1JIRJZqzj6qcRdQ6lsWSi1CwIh6EoKnCR+H7Npteai1CQEhc09C8PQI
3e/kpyHXJwSCqCcheGqFXN/XFIZcoxAIop6E4KmqTdIonvfgIu5bfUJaUXUkxA8utSTYiFXcDILHUstY4qm+oQ0I+pJCB5d/jHnsxlYST8G2
Ux+H+DvnG2IvCKUWUQ9CcGjayNwF+6Nqrq36MaZvGvIV1zH9YaodUIZRja1CMHzFgO1ydpER36NHHk+NHbXkIk6iOgB/2MsC7TVI2QAop6E4LFDe4
3SruEpU1pb4GvzXK360IepJCJpj2K9Vb/gmUfqSGlb3A7ACq3/IepJCJpj2K3VCd0rXKJRmpDByNyTEDQdNXvIrKRVmpDBiHoSgqa1Znm/uhcRjg5EPQLB01CzvELN8oSMRdSTEDS6gx8
UapYnZCzisUAImodYSi86cxLNtHjK/EWDDKFMIOpJCJpiFrOYc7m/WzNklChiNzT0IYjNIoq0CjLCGjEfUk
hIOXrrMPf1mEPUklDX2prsBQliIehLCQKcp5S6NsoSMRtSTEAb6ogN/LIO73w6inoQw2KRN0w0WN0QihjiHoSwsDvatupWlDFEPQlh4Lf31E1LK4QyhqgnIQz8buPtSA
uf5H9CXF1jLaPHp9FtCzDKFMPBr6f0Ai1ikpSVCGUJ6T0IY+FVPdbW0QihjiHoSwsDvatupWlohlDFEPQlh4Lf31E1LK4QyhqgnIQz8buPtSA
ct7RDKFKKehOC5iFd9SijHOxynpS1CGUItQhBk8U2LQ5919BLQmz+tpDekxA07TV5G+/A81q8bQplBlFPQtB01CZpMBO0yRLKAImDu4r04Hh
yPZ9dwAa+oMShRDZdaEG9JFq0ky18QrFl3hAO8xnfJyHtCMcy1JDyOAdi36szlhx2s5XNrGFfCvJLyaYDHdjEsri0cnSlI8f4kvsBK4FBtPAl
BZ6JDZCGJ2lNdJA97KKANUkOsSowk8FJnWFPMV1ZndKZrehEPc9myIf5idWst8mtyKm0JSeldgDM4Rug092TOkuxi90Usc71CRhDxYTjBXxlK
JHHGQnHRTzjIjOHKy3v4BKmRe+H/rR1keHGa2wFGjIeyOFTXkId+dRRj6rdKnm2qZtVjsLic4y6Wf2YgkSldqmHVU0LiYuVUkotUWdZ1uf06W
2qIzcut2lc+iG1RE1Q9ZKuAYWqoCaorUoppSbFpY5S21K6ComMUijUbN9yasXatSplGd+qR1Ubz1elqvrBd6tL+UKVS/K/kqXy1dqU6lqvhqs
sg7Ty6na10+dv6K1QqEkpn/+zelWd4/CLCw3lR5pKjDOU2OxyDRurTy1bsk9dFCszLeXfc4T2ClVFzVAD1CR1rbpSjT4iupf6yYfY5epY0w9q
ptb5aur3qqtJ5uJY7j+SvEm9q6cIB9Rjqn6Sj0FdtSJ2/qTYo/GMr6tQyskK5V89/TeuvamrJ6WUOqwme/wfnKRKfLY6kT8k9V+prGb5qm2Wq
hwn7Rj1nu/2l6g6CuVHPUV4TzWw+c261VNvm25GgTo1rpRf9bRX5SjUKHWqGqdy1VXqVPV0drSr946vbQNdeMcwJKzDQp9dvYa8Szvb3N/zh
C/pbuQwhvUMT+KMyrzHKabUW7lSS3s2atrz/6IWKQDZjOcpTyVX85G2WgHuTmL6IZuZXOirtguZGTdD+yj9fEkDmMcO3zIA+rGQ6lokOTOe9y
ynZzbSnU801vM6xcBxrAcmcQ6r2ZMNLGM6lX2KbstfE44foalPiVCd5xxWaq42zSbpJpcXmOx58eA+iyngJkzU1JY70WEBUshkDVJKGcn5nso
101prDW7xXHY0A3zXN4Cx0W/duNq3tBL+5FvGEdpwvzZZ1lThJSZbBlJdSg++0VjTAe4BYB1dgUlsI5sa2cAwh16Kd66keex7ey7VIBFOZohD
7r0hLDOP52lPUW4bx27heK6hkpZWPM9MDVJKGMWPGuTEc5unUvoeyAjXe+w/VdD0ergzOg1+vQZZd/OZBilHuJraGqUZackyLrHMeZV+mgN6T
WAjAG8wgm/Yy0M8wqxs4HdaxGfFqSQ9EgGbixPhRDprq8eekbzkYY1mmKUSO0tD/YpHNLyzYQcX8roGOYl051gPpbzYjG9PotYqXOOpXG8aJC
HVnrqcDRD9mzoHuZm7/DcnjhwGaZUXz3msoL1lzgNcErfu7Z/djOKx6PdDXEtL7mY8TzCnPPq2W3az+BZhv+cfU8EwzOzqWLoLn3uU64d8qnG
Ri7mBedYJoK6D2/4s07zBXg4lHB/kGz7iGdOicGIZdzOIA6xkLs+5mgMccPEIXs1yoNvDg/KxvskTuYvbk9i2cgNTOehaqovh+FDKUV46MIeq
VLD9f5ajqiFld4LBzQE2sIin+Y9DHYfZ49iGSgZzgQg9meZ4VmpkM9Gmz1vC7/mnJxlernYxX7KAZxJm435lypGv5cFi2ut0dntqQCKlF9co8
TamepRxAW8kHDdyLK3n3ejOOcxnkGOEEOuWONko5ZqcjJzH4iTbBS8yMulz7Pglkxzzs2nPHaaecSsPkt238x7kDfbxnAdZERpzMS+4ljL27O
ZygecazOxxGFJ2YqUhpSObk5T/Eb1dSjRhOPcZ0ppblvRHTaYz0DJnD79jnkcp/q42YOctcw2FvqTqmxOqoE2SP05jIef8xuPTlvAlVzLY8A7
P9XCmu3q6lx+ZzmWc6bk1f/Cgnvwu+GQa/+F++hqukf65p468bvNi/YGBJjUcKLKpxSud+dDTPMvRzT7TULOGh7PctrW8yD2AYhDnM563PLWk
C508lTvaMNrMe7n+yTCMT2yU01q6haucRD3ZMccirTVLaBl6SzIN4wyWl/6tc+9pCZdHzSb2M4epXOwx7NS1nkodbRjnp3SuXpdnKi/a9Dnfp
Rf/1ViXJ0Q9WXMDD1qkNmNwJuEUzcMi1RG1qOeY/wuGE431c5cnKawRVPJQSvFKf97nBJu9ZzktHdGaJ1GKN4hb2cLcpvQEfcC6fpqFF1jRwnU
49wlpNtspGIwvn1SZwXiCAn3jTlLaEqYx3lVuNi3jescRhw3HDJCZr97CNDY7b3dODcb1XT3j4Wkwl33YhamLUZDI56ni+Nzex1TpD1JM997D
Hwsq6Ju8xmIVpaI8VZ3u2xTmPuRrqy6K1IcXdO08Ex9znLA0EbmcAJ7pKHuminoweB7oa1oXd2MELPMgPSZ0TNMaVUj0vnWq2/aaDjGJ6SjJ7
```

```
eg79NcZui5oM7pyYwmiL92dV3va4mSOT+D8tUoZT05DivprpvPT9pGXqfkZ66Ln0dpG92FWCM3UYx1ec51OKTlrRx5CSjDlr8uyif4rKKRlsL
fuk9+TMk+xlmukqVeR1rmBGWlqUGl9Ftwx4IY8rLNPrcJKFgaWT0WiE4x3yPrSdBl/GZG5ykZzFi0x2MAtdwSrf63u5zGawlp6nV6oz0fK5zK
Elp5tMM73/X5NnC+fydYDyIxTYbywW9eTGDHbzmml9qhzTqc7jaWlRKvw5ibKraWljlGemxMNSs5MvjGcd8v7IINcBXjcGO1qtj2WJ77Wtcky
nNT/5lOKdIrbwqOfSy9yLpMyBUJwnP2hv/y+DO3fe5FyLCcgsHvO4ITb9zEnKkUoJIzy/k+d62F9gf4sXO84F7ecy0+S2GefVu6W2cyrJUJPb
NUjxzt9shrxmdvJ+gO1oxXKN+2etWea0o0TUkxfe5yzLHWv38UAZcM4/33FjtRWFDPa0relH/uChlP3s1EqXxeoVPOAqvY9LoIW/MSylLVqJj
Aj5SbmeDzyUKmGUT9fTblThFSYHOMZazvlOpiminryxlH6WayQ385gnhyvpYhVXMiCFW3g9I1xtj37iHE/7ymwWjfEyb3U3a11KVHd1Gvgyzb
mL1R56YvbU9u3jPTmKGep6bUu4ltma6jvAB7Z3yXgWBrK7dSPj6eW8tCJzT175jL7Mt7AMuY7qmjxipsI7/MU27zBf+1h2fpM/Wdh9lfIRI9j
iSZJxL38p7quJB+hKb1rxC51sLaHct3XsYBKTyEpiA0gNk3JorGnt0ys7GMQnDtduG1doHNhtpzfteM1kNhLhNFYylKVJylzKnbZ5JWz0suYo
6sk7aziD92liSr+UamnbfLrd9+K5PfeSZ+kOsJj3+RezPfrvLO9gmfWhh/N/ZR7zgBn0MjlIieDVmln53OgeNmu5lNkW4xvFKp7nSe3DunV04
Z82jiQbsJibeCQpeTv835syuEuGTfS0dGA6yObBKdsoLmedRfp2ruINz86FJ9j4YlX8g4+TaM1h21BHadhsERJzuMMi9RC3MTWQOac9DGeMjW
+2yH68kLcRiXpKjq2c6jobcvSwhwtMXqmgCXM89xbr2zry7cnYJP2n27kOOXrVE/zFwrquArNs3B/q4Al68q1N3jA+ddkDoBlRT8lSwBksT3c
jQmMTl1hYb5/Cix4XBIbbKLL5KcT4sNsRVpi0pLLE1RY7PKswL0DfGZ9zsu2Uezs+8+9kzjvWc08DU3J6uiMWMEjfRsrEaMF+1l6cJSfDTvoz
2/N2x7LOAm618N4wmIc9bNrtyr2W6Sql4Ah25p3m/l0QODs6DrLeIaygsSG1DvPpEZipaCEXMp4HLLVDdd7gfv6o9Vm0pTzwk8n5rrsfQivej
Kkn42WbyDiPMozv2sTtmEa54zy7sjVGTNnta+y+i4HM4hwfEnRxadLvsvOSjjr3EHkMM6WO4xv+7nheJV62DHt9mFtS6H9WtI2baL80fSN/TL
qeCGZrNrfA4cHxA4NZYrqSLZhHbw0WXdYoJvMJr9o4YLyNrlziuio8MOl+7WheSUwoD3yqKdpDaViCTw2rNTVNG0m9ktixNYa1qZGyr0C/WwH
2MZiXHcNchUOFpK9AKm/cq2lNnin1ETY7erYcZxPf7syUVnSm2IY7s1dPa7X5kvw+fFdscXzOlbxsSu3MawzyMQpw4xPymG7zEu7HFwx1ecmU
T/rqm8KcZYNRY6WIiutzvaolaCRg+KfM8hCdIxW5qVBMfpnaFBxhc0pbSPcxxOJdmc3LDqG86tlsBFmR4nLzuTbp+x0mIt7X5gxF3x2dGq9YD
pP783SguxZ2MJA/2kzVHMcSRmutrcg8y5YNvKRlLWpa3Hz/Ok3BslcYpui2atqEu8HFT5A3DnNZwKHU9ZOc5UopW8i32HxQhbdsgz/dTjXL9K
kptsDuMXQy7jvkEnvGK3scjF/DYqJllMIRll5d9VHCvZxlE7w1hyd41nL4nhqPm3e2ZhPZAuru89CZr7kx4Xic7eKkd4q4wvTOuoM1vuX+ygg
PLmi9oPh9wLeHXlbxj5TPXcwEi9SGzDX5b4ykW79Z/5tyv9Vuatroci6RfyXpgM6aa7RHV04exeWsski/yfO8bqospJOtAe1IlmoKUb/FFCKL
I4YFqznb1z9gBf0N6yc76OuzT7aNfha+ZvbQz+e8UQHnaAwjrbhFU6Ds4NnEIF+D479Zuj/pwEyLFZ5xNuHbK6e8qrvTJt3ZH5FimM+B/AFGa
pgK0MFeLrCcOZzCxQHXvJ2+3G+Tl8fntgNv7/zIQKtN90fsnj6mHVNc48hasY3x9LLY9rmFLvxvimP/Qh6kvY0SKeA0fp90iMMIRTxKW09bKZ
LhHst+RWaxn8fo6rA51xtjLCdD+8cCUJeSbyOhtqP3JyfszAfcXoIHGMbFKfa5i3mF9kkEBw2aLQy0jDM8nb4B13yY2xlgs3pZk7n8yYcF5SG
m09lyfwJZCaOnHLrThDoexZZQWEa+cJw0zKIzJ1AvicYXsJnlHt7yHWhDvST2DO5gC5/arHNU5ypDyjNJqupBhh3ty1z7eBUZY0h5zYP6MNbj
hUNsZJnF7xluUBXubYZGzKK7RfqtBtcnh2zNNmulaKc0zcaH56kee9MtyKOeKZCDPUVstbxq1tQ1+U1wv4O6G67kVl7zUFMLetPSdN/vSujZj
jH41Fxg8g2RxxkJx0W2W4biOYa+NLN55hbSwNb0w54SvuVj256xQT0JghvZ9KUNNQw3qUroe9e2tYn5kpNSrPfPNiuB7a3fu8LRgHgsEJKjhP
d4z6WMvQve1I1Y7OyOOol6OnqRPXeCfuzi9x7yMY9j51vKbC4qHDWIehJ0047LbHJeZVvKUu3MScP1YimEiqgnQTcP29xVKql4MUa+s5lA1bW
PQMhARD0JejnTdqv0vz34FrenhHmW6encCycEjKgnQS928044ei73grW92hc+pQoZjKgnQS92hgMrPATsdMbKxHcFL/mUKmQwop4EvdS3Sfe/
9+0jFpnSvtLo+lDIOEQ9CXqpaJPuxSLamV8sekqh+GwU0oWoJ0Ev1j7El2iJEldoSklll6hQZhCrcUEvl1rOEd2jRXahKSWZKC1XkM/xVKM8
B/iB5Twct8V1Cn1YZOk9vT7XcjpNqcRhdrOGZ3k3LvftuAANxWzjVVMPrwWzAJjBQ4acVcB4iwGrmU6MIo9jKcdhtrOCv7MhlteItxPKboj5L
ziFpxJySn/f3ZyfkFPatiPSCukf3Z/ahylAp1jZLK5mCC2oTAl7Wc+TzI/lVWMJcHV0M39D5gE3xgULnciFfM+A6NFThngzT1h4T1PykY/Ozz
nKyLOqkybZXU2yJ3o+9ybTucvjcmcrpWZbnJWjNprOuzAuf7Mp92yDhFuj6ZtVliFHKaUu8NDy1qrYUEeRahzLbWrIWxXL6W3IKf190ww5Uy2
kXR9NuUAppeJac73pF3eO5eUqpZTqHTteo5R6OnaUrb5XSv0tdrzYIGeS+bfL4E7QSx/D8X5usnSjlgqFphSvvhqzuRVYzBm0ohmdeB7o4sG5
y2mcANxMF5rRhrNYCyZHew+TRx55XMkh4ExDbj6wCDiebh7bauQKKvAz+XSgGR24lL1UM4WmuCTahjyT76fTYzmJfcPFsfQ8U7/uO+B2m2s7B
FjNCWSTRTv2Av1tWz4bGEKF6FFPGkbTSnkxrhUWnmdlcCfopZ/h+EV+1ibbPJTz6qmxEbWBSTHbqamcBNR0CKMQoTOWLfb4rudppph2+W2Lqt
9VjKGrwXdoCzoDD1KDzuSn6EgxD5gTW1pYywgGmHy8b7B9BayxiZeyy+Gl8RgP0YDrmGKRVwv4gE0AfMV02jgMsGdxJzXpFx3+DcEc2LzA+dU
lvSdBJzVMD++/NEovNKV4jZYb8WFWOte0kk508hAWogEkOIfZCdS1ealXojEYLOPzgV94n5lAfophC5qQGI2mAGiYkiSvLGQDcAdVLfIOAB3I
jR6NobeD//9VfMcR14RZDAXeTG6tVXpPgk5OM7zwNvgO2RXPAYoNDuVacJ2ncBSRsw4COXFO0zaYne8bqE1iny2yUlg1QVH+LyOBLJpSnQUGt
275wFscZCb3cxzdUroalSEhKuMeMIWZmBJr5RyeTsiZEduV+DRz4tL7xPVbjBP0JfyFZ6nNWIsQEB/RlT78ws/s4D9s5j1mYu807g0mMITrOE
g3joPoMkEpw+PC2V4cN+EfRdSToJPehmOr6CKpM9LC2+WjfOXBPXNpeNdGcdbreZ5mxdw8NtaiClCeKkAu9eJCY0aGdjOBb/iCzgxLWVnHB4I
4hHnU0zv2bbMhZ0Ds2+KE9Mo0jftu5AXupAU3WQRavZvynEkzalObE4FraGw5CIzwOhOiw7shQFHcKl6E6rGZKaw8mYp6EnRinBj2u5ElkVss
0iowk25J+J4v5C4glxswx442U0LiMxJ5kBP7XLdFg2NV5RUG8Ne4uM35QBELAHidzuQzPkUr9/iW5mK295rAd9Fv3xhyRsTiACb6W5/rGF/6E
BOZQW3GmiLh7OIGAOpSk6ZMph2DHNTTJ/xAQwZH1dNbHDDkP+4cZ0bUk6CPWnQ0pOh0d1KL1pbp9XibHi4BsyOB0mqzmUImAZ24ATyoih0Qm2
```

```
cBqA7sMT1kR+pYzoCEWMb5QIVocMnqQEN6ssS1TiOFHE/tuOMqmOPWLLLtB85NOpR4hJe5nXZMiCqjUh6mDs+xiAIK2MgbtKOWg5QS3mQ053I
CJ5BCX1qmxgV99DNN/vqNnxhPF9ucNsxzCZgdWWkqXQLvBeBhTXEjcEKc6WU/cLB/b0v8E9WEzkAhueSSSza/Ar9zrdG6DT1ig6AqdOfI7wmS
Eu4E6jLCkH42V8T1kRvhZhr7OnA8NwH7DQakHpDek6APswWMnnCndtJL6c6XPMsCltn0iApZTG/u4yq2cpj6tAd+iA2IjsjfHPt+SXSW6B0Ok
cM3rOMXKtOEY8H0kI3lAqAK9TkO4sKf5QP7aRVT0fdwJxdxQ0ILn4xFTS4y9TyP8Cb5dGYb31JELq2oAcy1KeudXnFzUfMtoyD/m+V0NcWwW0
V7JtCBnznIcZyJm0ubRfxCTa4G3rZYiBgaZ4/+msVMl2erW/nIx+1jtqE+XaP0t03SzaxX59uc3Vx9mFDyv2pAXO5sg5zesZzL1A9x6cXqVVX
N4RevVi1jecuUUq/Hle2glFKqT+w4kULb352tpqp9cSV3qrviciN23p0szotYjeda5BitxqfZSOsfKxF/HdclnPu5ahTLM1qNRz7ToyVHGNKN
VuNTzS2VQFKCLhpahGk8OzoxrIOvaOOp3IOWU+gA9WlJDRS72WboOTUzDA43xQ1Ly9GcxtSgiEI2xiaaI7RNWG8qSPClfhJZbGe7IWVbzIapU
4Kkw46hQivRmtpUZjff821CnzRiKGFlIlGVlsAaC0ujJob5op2x4J5GaR2jg9VVCeWbchy5HKSI7xJ2WJajA4nXDqAujQFYbwhE39JgV1Vg9k
Qv6knQRQ+WmtLut4lOlwrfxS2FO5drrq1OIa2IehJ0cbJF2Pk91E8wKfRDOy6nKXsoYBs1OI+uNlbY62ivqUYhzYh6EnTR1DBgitCN5QHVV59
zaU0jGlCTXCpQzM/sZBvvMjOgGoWQEfUk6CKHfZQzpTa3VFqC4AExLBB0Ucy1nEtbmlOJQxRRxHY2xSZdBSFp/h9DXBZ6srN36QAAAABJRU5E
rkJggg==
```

```
img.save("pics/geeks.png")
```

Looping through csv file

Looping through rows in a dataframe: https://cmdlinetips.com/2018/12/how-to-loop-through-pandas-rows-or-how-to-iterate-over-pandas-rows/ (https://cmdlinetips.com/2018/12/how-to-loop-through-pandas-rows-or-how-to-iterate-over-pandas-rows/)

In [2]:
```
df = pd.read_csv('data/data_clean.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	game_id	game_title	month	peak_users	picture_url
0	0	920120	FLIP FLAPPERS: Pure Storage	2020-05-01	0	https://steamcdn-a.akamaihd.net/steam/apps/920...
1	1	919890	Navyblue and the Spectrum Killers	2020-05-01	0	https://steamcdn-a.akamaihd.net/steam/apps/919...
2	2	919890	Navyblue and the Spectrum Killers	2019-03-01	1	https://steamcdn-a.akamaihd.net/steam/apps/919...
3	3	919890	Navyblue and the Spectrum Killers	2019-02-01	1	https://steamcdn-a.akamaihd.net/steam/apps/919...
4	4	919890	Navyblue and the Spectrum Killers	2019-01-01	1	https://steamcdn-a.akamaihd.net/steam/apps/919...

TODO: Just need a distinct row of each game

In [9]:
```
for index, row in df.head(10).iterrows(): #exclude the .head(10)
    # access data using column names
    print(row['game_id'], row['picture_url'])
```

```
920120 https://steamcdn-a.akamaihd.net/steam/apps/920120/logo.png
919890 https://steamcdn-a.akamaihd.net/steam/apps/919890/logo.png
919890 https://steamcdn-a.akamaihd.net/steam/apps/919890/logo.png
919890 https://steamcdn-a.akamaihd.net/steam/apps/919890/logo.png
919890 https://steamcdn-a.akamaihd.net/steam/apps/919890/logo.png
919890 https://steamcdn-a.akamaihd.net/steam/apps/919890/logo.png
919890 https://steamcdn-a.akamaihd.net/steam/apps/919890/logo.png
919890 https://steamcdn-a.akamaihd.net/steam/apps/919890/logo.png
919890 https://steamcdn-a.akamaihd.net/steam/apps/919890/logo.png
919670 https://steamcdn-a.akamaihd.net/steam/apps/919670/logo.png
```

Since there is duplicated, should only download each picture once

Unique game ids

In [6]:
```
df
```

Out[6]:

	Unnamed: 0	game_id	game_title	month	peak_users	picture_url
0	0	920120	FLIP FLAPPERS: Pure Storage	2020-05-01	0	https://steamcdn-a.akamaihd.net/steam/apps/920...
1	1	919890	Navyblue and the Spectrum Killers	2020-05-01	0	https://steamcdn-a.akamaihd.net/steam/apps/919...
2	2	919890	Navyblue and the Spectrum Killers	2019-03-01	1	https://steamcdn-a.akamaihd.net/steam/apps/919...
3	3	919890	Navyblue and the Spectrum Killers	2019-02-01	1	https://steamcdn-a.akamaihd.net/steam/apps/919...
4	4	919890	Navyblue and the Spectrum Killers	2019-01-01	1	https://steamcdn-a.akamaihd.net/steam/apps/919...
...	...	...	...	...	...	...
696440	696440	960170	DJMAX RESPECT V	2020-04-01	1736	https://steamcdn-a.akamaihd.net/steam/apps/960...
696441	696441	960170	DJMAX RESPECT V	2020-03-01	2777	https://steamcdn-a.akamaihd.net/steam/apps/960...
696442	696442	960170	DJMAX RESPECT V	2020-02-01	1095	https://steamcdn-a.akamaihd.net/steam/apps/960...
696443	696443	960170	DJMAX RESPECT V	2020-01-01	1762	https://steamcdn-a.akamaihd.net/steam/apps/960...
696444	696444	960170	DJMAX RESPECT V	2019-12-01	2290	https://steamcdn-a.akamaihd.net/steam/apps/960...

696445 rows × 6 columns

In [9]:
```
# df.groupby('month').peak_users.nlargest(10).reset_index()
monthly_top10_df = df.sort_values(['month', 'peak_users'], ascending=False).groupby('month').head(10)
```

In [15]:
```
monthly_top10_df.to_csv('data/monthly_top10.csv', index=False)
```

In [10]:
```
game_id_series = monthly_top10_df.game_id
```

In [11]:
```
game_id_series.head()
```

Out[11]:
```
0    920120
1    919890
2    919890
3    919890
4    919890
Name: game_id, dtype: int64
```

```
In [11]: # total game ids
         len(game_id_series)

Out[11]: 950
```

```
In [12]: # itemized list of game IDs
         unique_game_id_list = list(game_id_series.unique())
         len(unique_game_id_list)

Out[12]: 115
```

```
In [13]: unique_game_id_list[:10]

Out[13]: [730, 570, 578080, 105600, 271590, 359550, 1100600, 582010, 252490, 346110]
```

Test interation of images

```
In [14]: for game_id in unique_game_id_list:
             pic_url =  f'https://steamcdn-a.akamaihd.net/steam/apps/{game_id}/logo.png'
             r = requests.get(pic_url)

             if r.ok:
                 img = Image.open(BytesIO(r.content)) #If there is an image online, read in the data as bytes
             else:
                 img = Image.open("steam-logo-default-small.png") #If there isn't an image online, save a default imag
         e

             img.save(f"pics/{game_id}.png")
```

```
In [18]: last_id = 1324480
         last_id_list = unique_game_id_list.index(last_id)
         last_id_list

Out[18]: 699
```

TODO: Just download pictures in top ten

In [17]:
```
cont_list = 1324480

for game_id in cont_list:
    pic_url =  f'https://steamcdn-a.akamaihd.net/steam/apps/{game_id}/logo.png'
    r = requests.get(pic_url)

    if r.ok:
        img = Image.open(BytesIO(r.content)) #If there is an image online, read in the data as bytes
    else:
        img = Image.open("steam-logo-default-small.png") #If there isn't an image online, save a default imag
e

    img.save(f"pics/{game_id}.png")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-17-b0aa3638e50c> in <module>
      4 
      5     if r.ok:
----> 6         img = Image.open(BytesIO(r.content)) #If there is an image online, read in the data as bytes
      7     else:
      8         img = Image.open("steam-logo-default-small.png") #If there isn't an image online, save a defa
ult image

c:\users\youth\appdata\local\programs\python\python37-32\lib\site-packages\PIL\Image.py in open(fp, mode)
   2670         return None
   2671 
-> 2672     im = _open_core(fp, filename, prefix)
   2673 
   2674     if im is None:

c:\users\youth\appdata\local\programs\python\python37-32\lib\site-packages\PIL\Image.py in _open_core(fp, fil
ename, prefix)
   2656                 elif result:
   2657                     fp.seek(0)
-> 2658                     im = factory(fp, filename)
   2659                     _decompression_bomb_check(im.size)
   2660                     return im

c:\users\youth\appdata\local\programs\python\python37-32\lib\site-packages\PIL\ImageFile.py in __init__(self,
fp, filename)
    101 
    102         try:
--> 103             self._open()
    104         except (IndexError,  # end of data
    105                 TypeError,  # end of data (ord)

c:\users\youth\appdata\local\programs\python\python37-32\lib\site-packages\PIL\PngImagePlugin.py in _open(sel
f)
    576 
    577             try:
--> 578                 s = self.png.call(cid, pos, length)
    579             except EOFError:
    580                 break

c:\users\youth\appdata\local\programs\python\python37-32\lib\site-packages\PIL\PngImagePlugin.py in call(sel
f, cid, pos, length)
```

```
    138 
    139         logger.debug("STREAM %r %s %s", cid, pos, length)
--> 140         return getattr(self, "chunk_" + cid.decode('ascii'))(pos, length)
    141 
    142     def crc(self, cid, data):

c:\users\youth\appdata\local\programs\python\python37-32\lib\site-packages\PIL\PngImagePlugin.py in chunk_zTX
t(self, pos, length)
    467                                       comp_method)
    468         try:
--> 469             v = _safe_zlib_decompress(v[1:])
    470         except ValueError:
    471             if ImageFile.LOAD_TRUNCATED_IMAGES:

c:\users\youth\appdata\local\programs\python\python37-32\lib\site-packages\PIL\PngImagePlugin.py in _safe_zli
b_decompress(s)
     84     plaintext = dobj.decompress(s, MAX_TEXT_CHUNK)
     85     if dobj.unconsumed_tail:
---> 86         raise ValueError("Decompressed Data Too Large")
     87     return plaintext
     88 

ValueError: Decompressed Data Too Large
```

```
In [16]: img
```

Out[16]:

Looping over subset df

```
In [17]: df = pd.read_csv('monthly_top10.csv')
```

In [18]:
```
subset_df = df[['game_id', 'game_title']]
subset_df
```

Out[18]:

	game_id	game_title
0	730	Counter-Strike: Global Offensive
1	570	Dota 2
2	578080	PLAYERUNKNOWN'S BATTLEGROUNDS
3	105600	Terraria
4	271590	Grand Theft Auto V
...	...	...
945	240	Counter-Strike: Source
946	72850	The Elder Scrolls V: Skyrim
947	42690	Call of Duty: Modern Warfare 3 - Multiplayer
948	8980	Borderlands GOTY
949	550	Left 4 Dead 2

950 rows × 2 columns

In [19]:
```
subset_df.drop_duplicates(inplace = True)
```

```
c:\users\youth\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#r
eturning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [20]:
```
subset_df.count()
```

Out[20]:
```
game_id       115
game_title    115
dtype: int64
```

```
In [21]: subset_df
```

```
Out[21]:
        game_id                                       game_title
   0        730                   Counter-Strike: Global Offensive
   1        570                                           Dota 2
   2     578080                    PLAYERUNKNOWN'S BATTLEGROUNDS
   3     105600                                         Terraria
   4     271590                               Grand Theft Auto V
 ...        ...                                              ...
 909      42690   Call of Duty: Modern Warfare 3 - Multiplayer
 912     200510                              XCOM: Enemy Unknown
 917      71270                             Football Manager 2012
 926     200710                                     Torchlight II
 948       8980                                  Borderlands GOTY

115 rows × 2 columns
```