1 Using list comprehensions

The code to get **col** can be reduced using a list comprehension. Going from this :

```
col = []
i = 0

# For each row, store each first element (header) and an empty list
for t in tr_elements[0]:
  i += 1
  name = t.text_content()
  print(str(i) + name)
  col.append((name, []))
```

to this :

```
col = [(t.text_content(), []) for t in tr_elements[0]]
```

2 Removing a variable

From :

```
for j in range(1, len(tr_elements)):
    # T is our j'th row
    T = tr_elements[j]
    # If row is not of size 10, the //tr data is not from our table
    if len(T) != 10:
      break
```

to :

```
for elt in tr_elements[1:]:
    # If row is not of size 10, the //tr data is not from our table
    if len(elt) != 10:
      break
```

3 Removing try/except pattern

Using **try** and **except** should be avoided when a simple test does the job, we go from this :

```
# Convert any numerical value to integers
try:
  data = int(data)
except:
  pass
```

to this :

```
if type(data) == float:
    data = int(data)
```

Also, if **except** was to be kept, it should catch named errors, for instance :

```
except TypeError:
  pass
```

4 Removing manual variable incrementation using enumerate

From :

```
i = 0
# Iterate through each element of the row
for t in T.iterchildren():
  data = t.text_content()
  # Check if row is empty
  if i > 0:
    # Convert any numerical value to integers
    try:
      data = int(data)
    except:
      pass
  # Append the data to the empty list of the i'th column
  col[i][1].append(data)

  # Increment i for the next column
  i += 1
```

to :

```
# Iterate through each element of the row
for ind, t in enumerate(elt.iterchildren()):
  data = t.text_content()

  # Convert any numerical value to integers
  if ind > 0 and type(data) == float:
    data = int(data)

    # Append the data to the empty list of the i'th column
  col[ind][1].append(data)
```

5 Changing variable names

Variable names should reflect what they contain. Also, starting a variable name with an uppercase should be avoided as they are normally used for class names in python.

From :

```
Dict = {title: column for (title, column) in col}
df = pd.DataFrame(Dict)
```

to :

```
dict_pokedex = {title: column for (title, column) in col}
df = pd.DataFrame(dict_pokedex)
```

Final code

```python
import requests
import lxml.html as lh
import pandas as pd

if __name__ == '__main__':
  url='http://pokemondb.net/pokedex/all'
  #Create a handle, page, to handle the contents of the website
  page = requests.get(url)

  #Store the contents of the website under doc
  doc = lh.fromstring(page.content)

  #Parse data that are stored between <tr>..</tr> of HTML
  tr_elements = doc.xpath('//tr')

  # For each row, store each first element (header) and an empty list
  col = [(header.text_content(), []) for header in tr_elements[0]]

  # Since out first row is the header, data is stored on the second row onwards
  for elt in tr_elements[1:]:

    # If row is not of size 10, the //tr data is not from our table
    if len(elt) != 10:
      break

    # Iterate through each element of the row
    for ind, t in enumerate(elt.iterchildren()):
      data = t.text_content()

      # Convert any numerical value to integers
      if ind > 0 and type(data) == float:
        data = int(data)

      # Append the data to the empty list of the i'th column
      col[ind][1].append(data)

  dict_pokedex = {title: column for (title, column) in col}
  df = pd.DataFrame(dict_pokedex)
```