Practica Matplotlib

Nombre: Celic Gabriel Hernández Archundia

Matrícula: 2877240

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv("pokemon.csv")
df.head()
```

Out[2]:

	abilities	against_bug	against_dark	against_dragon	against_electric	against_fairy	against_fight	against_fire	against_flying	against_ghost
0	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0	2.0	1.0
1	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0	2.0	1.0
2	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0	2.0	1.0
3	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5	1.0	1.0
4	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5	1.0	1.0

5 rows × 41 columns

In [3]:
```
columnas = list(df.columns) # Traer todas las columnas a una variable, df con muchos datos
columnas
```

Out[3]:
```
['abilities',
 'against_bug',
 'against_dark',
 'against_dragon',
 'against_electric',
 'against_fairy',
 'against_fight',
 'against_fire',
 'against_flying',
 'against_ghost',
 'against_grass',
 'against_ground',
 'against_ice',
 'against_normal',
 'against_poison',
 'against_psychic',
 'against_rock',
 'against_steel',
 'against_water',
 'attack',
 'base_egg_steps',
 'base_happiness',
 'base_total',
 'capture_rate',
 'classfication',
 'defense',
 'experience_growth',
 'height_m',
 'hp',
 'japanese_name',
 'name',
 'percentage_male',
 'pokedex_number',
 'sp_attack',
 'sp_defense',
 'speed',
 'type1',
 'type2',
 'weight_kg',
 'generation',
 'is_legendary']
```

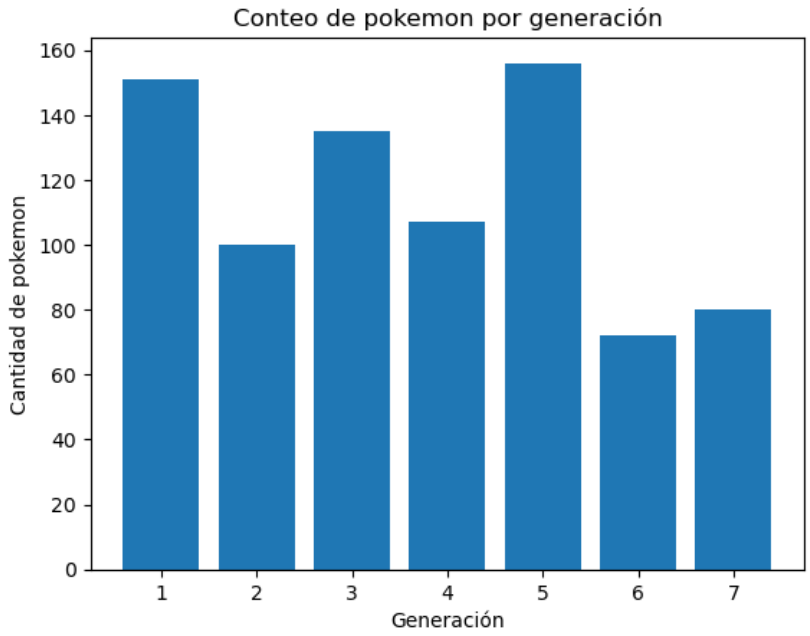
Realizar una visualización por ejercicio que respondan cada cuestionamiento:

1. ¿Cuántos nuevos pokemon hay por generación?
2. ¿Cuántos pokemon legendarios hay por generación?
3. Visualización por peso y altura.
4. ¿Cuál es el tipo más común de pokemon?
5. ¿Cuáles son las combinaciones de tipo más comunes?
6. ¿Qué pokemon son los mejores en términos de sus stats? (attack, defense, sp_attack, sp_defense, speed, hp)
7. ¿Cómo cambiaron los stats en promedio conforme avanzaban las generaciones?
8. ¿Un pokemon es más fuerte entre más difícil es de capturar?

Ejercicio 1. ¿Cuántos nuevos pokemon hay por generación?

In [4]:
```
import matplotlib.pyplot as plt
df2 = df[ ["name","generation"] ].groupby("generation").count() # Cuento todas las entradas. Devuelvo un conteo de todo

plt.bar(df2.index, df2["name"])
plt.title("Conteo de pokemon por generación")
plt.xlabel("Generación")
plt.ylabel("Cantidad de pokemon")
plt.show()
```



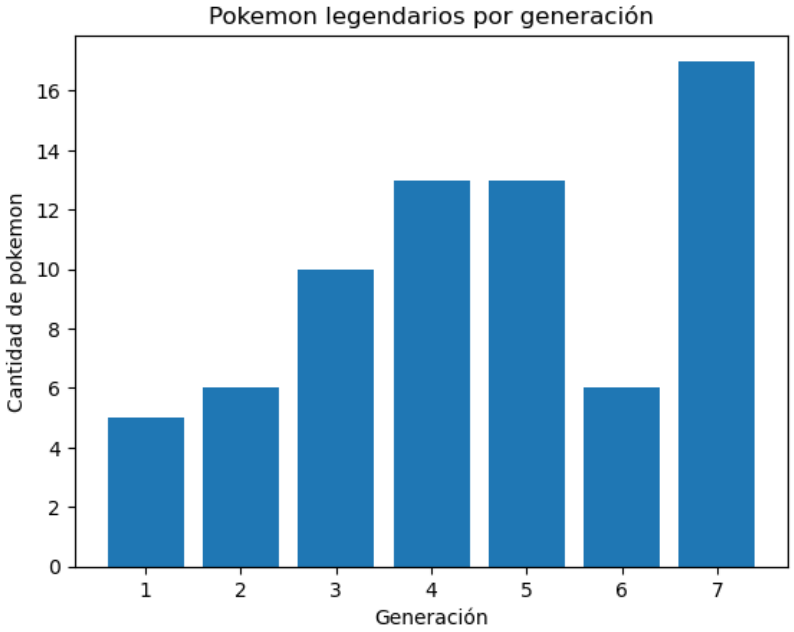
Conclusión 1

En esta gráfica de barras, correspondiente al conteo de pokemon por generación, se puede observar que la quinta generación tuvo más cantidad de pokemon que en las demás. Seguido de la primera generación.

Ejercicio 2. ¿Cuántos pokemon legendarios hay por generación?

In [5]:
```
df3 = df[ df["is_legendary"] == 1][ ["name", "generation", "is_legendary"] ].groupby("generation").count()

plt.bar(df3.index, df3["name"])
plt.title("Pokemon legendarios por generación")
plt.xlabel("Generación")
plt.ylabel("Cantidad de pokemon")
plt.show()
df3
```



Out[5]:

generation	name	is_legendary
1	5	5
2	6	6
3	10	10
4	13	13
5	13	13
6	6	6
7	17	17

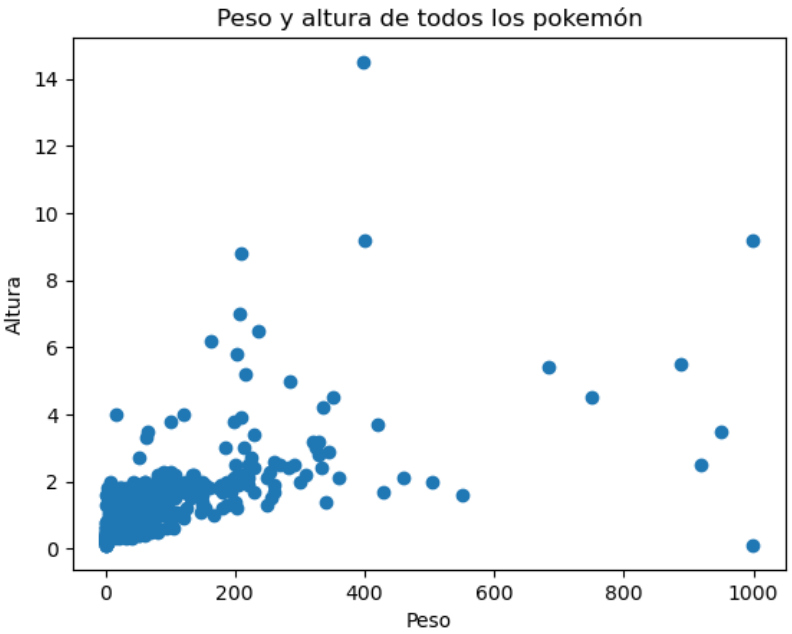
Conclusión

En la gráfica de barras anterior se puede observar que la séptima generación fue en la que más pokemon legendarios aparecieron; sobrepasando a los vistos en la cuarta y quinta generación.

Ejercicio 3. Visualización por peso y altura.

In [6]:
```
df4 = df[ ["weight_kg", "height_m"] ]

plt.scatter(df4["weight_kg"], df4["height_m"])
plt.title("Peso y altura de todos los pokemón")
plt.xlabel("Peso")
plt.ylabel("Altura")
plt.show()
```



Conclusión

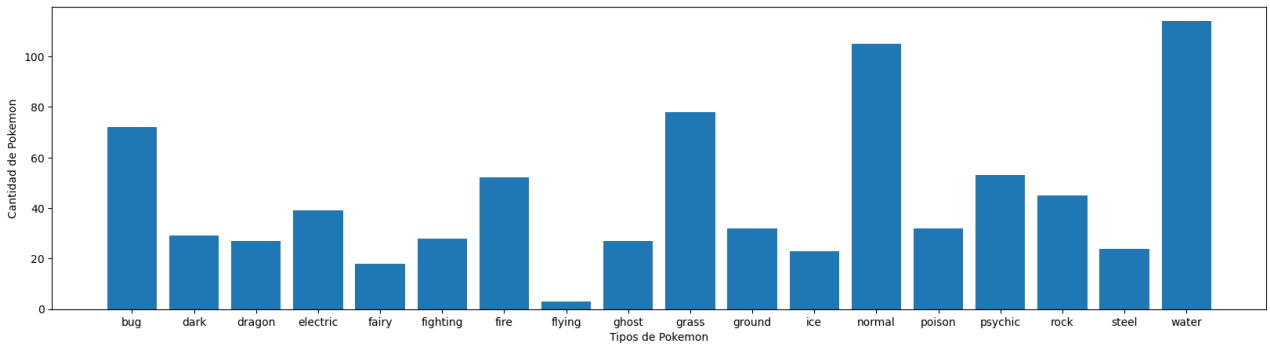
En el gráfico de puntos anterior se puede observar que la gran mayoría de los pokemones comparten una altura y peso de entre menos de un metro hasta 2 metros, aunado a un peso inferior a los 100 kilos. También, pueden observarse ciertas anomalías; por ejemplo, un pokemon con una altura menor a un metro, con un pero descomunal de una tonelada; o bien, algunos pokemones con alturas superiores a los 8 metros, junto con pesos iguales o mayores a los 200 o 400 kilos.

Ejercicio 4. ¿Cuál es el tipo más común de pokemon?

In [19]:
```
df5 = df[ ["name","type1","generation"] ].groupby("type1").count() # Contar
df5

plt.figure(figsize=(20,5))
plt.bar(df5.index, df5["name"])

plt.xlabel("Tipos de Pokemon")
plt.ylabel("Cantidad de Pokemon")
plt.show()
```



Conclusión

En la gráfica de barras se puede apreciar que, los tipos más comúnes de pokemon, son los de tipo agua (mayor a 110); seguido de los de tipo normal (mayor a 100); seguido de los de tipo insecto (mayor a 70). Siendo los de tipo agua claramente los más abundantes.

Ejercicio 5. ¿Cuáles son las combinaciones de tipo más comunes?

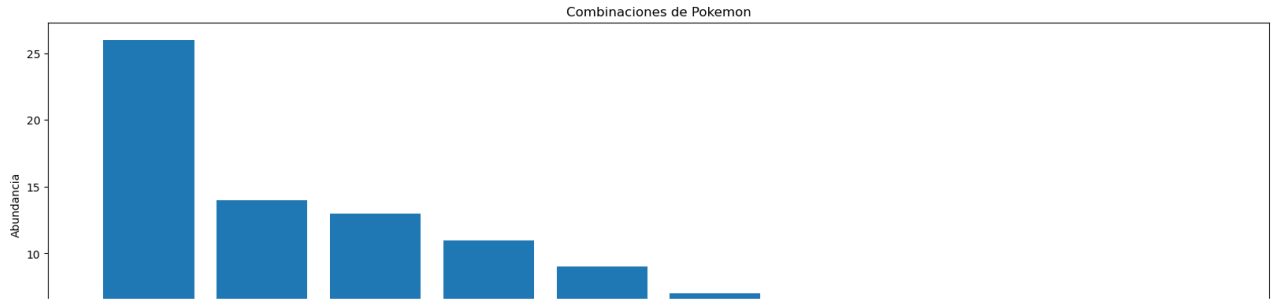
In [20]:
```
df6 = df[ ~df["type2"].isna() ]

df6.loc[:, "type"] = df6.loc[:, "type1"] + "-" + df6.loc[:, "type2"]
df7 = df6[["name", "type"]].groupby("type").count().sort_values("name", ascending=False)

plt.figure(figsize=(20,6))
plt.bar(df7.index[:10], df7["name"][:10])
plt.title("Combinaciones de Pokemon")
plt.xlabel("Combinaciones")
plt.ylabel("Abundancia")
plt.show()
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df6.loc[:, "type"] = df6.loc[:, "type1"] + "-" + df6.loc[:, "type2"]
```



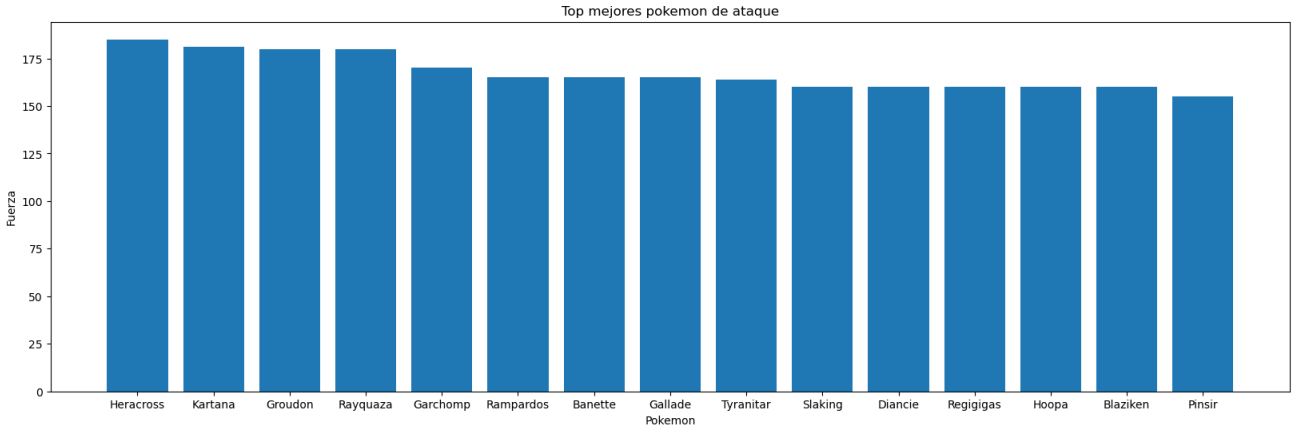
Conclusión

En la gráfica anterior se puede ver, con una clara diferencia, que las combinaciones más comunes de pokemon son las de las parejas de tipo normal y volador; de tipo pasto y veneno; y de tipo insecto y volador. Siendo la pareja de tipo normal y volador por creces la mayor de todas (casi el doble que las de tipo pasto y veneno; o la de tipo insecto y volador).

Ejercicio 6. ¿Qué pokemon son los mejores en términos de sus stats? (attack, defense, sp_attack, sp_defense, speed, hp)

In [23]:
```
df8 = df[ ["name", "attack"]].sort_values("attack", ascending = False)
plt.figure(figsize=(20,6))

plt.bar(df8["name"][:15], df8["attack"][:15])
plt.title("Top mejores pokemon de ataque")
plt.xlabel("Pokemon")
plt.ylabel("Fuerza")
plt.show()
```

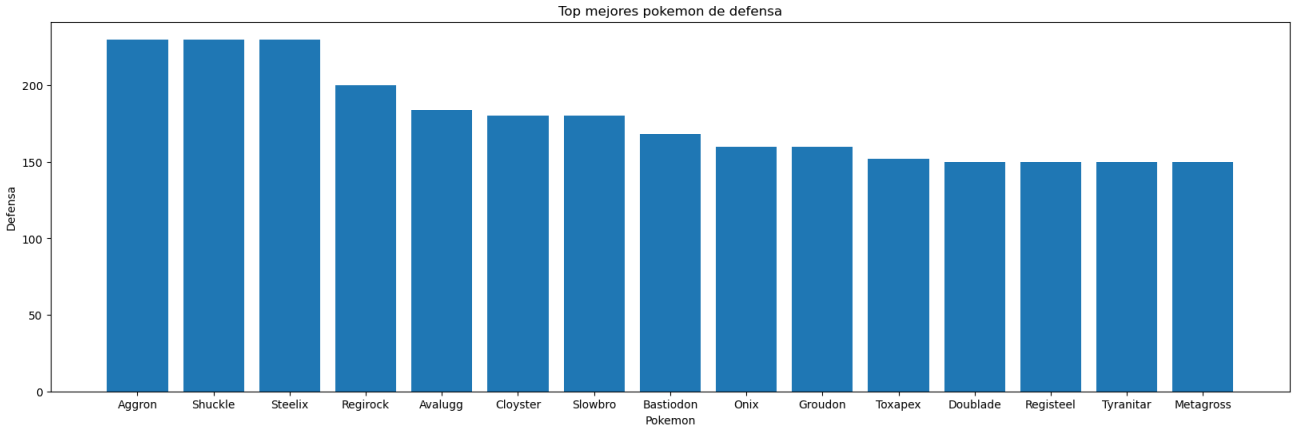


Conclusión

En el gráfico de mejores pokemon de ataque, se puede concluir que los mejores pokemon son Heracross, Kartana y Groudon. Siendo Heracross el pokemon con mejores stats de ataque (con casi 200 puntos)

In [26]:
```
df8 = df[ ["name", "defense"]].sort_values("defense", ascending = False)
plt.figure(figsize=(20,6))

plt.bar(df8["name"][:15], df8["defense"][:15])
plt.title("Top mejores pokemon de defensa")
plt.xlabel("Pokemon")
plt.ylabel("Defensa")
plt.show()
```

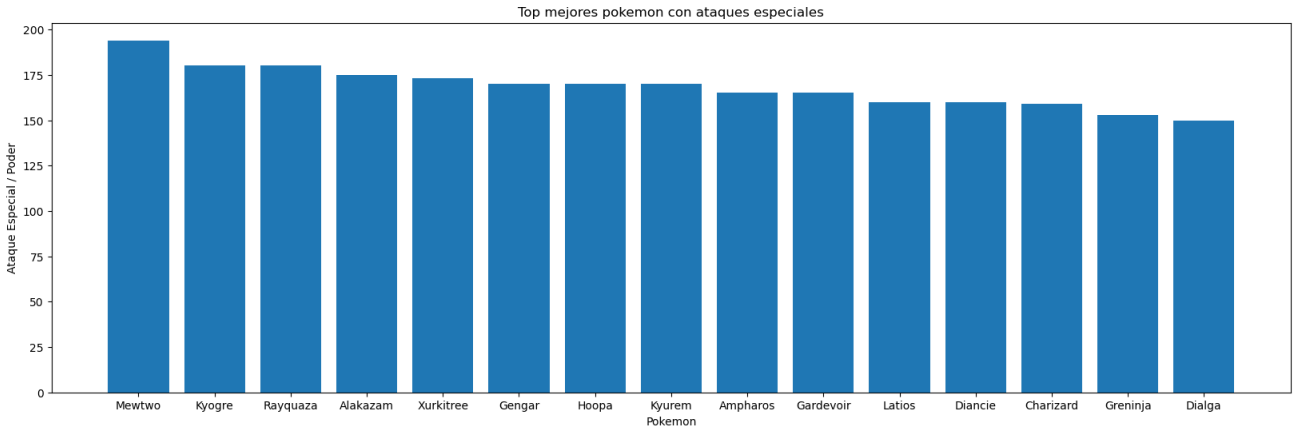


Conclusión

En el gráfico de mejores pokemon de defensa, se puede observar que los mejores pokemon son Aggron, Shuckle y Steelix. En esta ocasión, los 3 pokemon compartirán el primer lugar en las stats de defensa; ya que, según la gráfica, los tres pokemon comparten stats similares (alrededor de 250 puntos).

In [28]:
```
df8 = df[ ["name", "sp_attack"]].sort_values("sp_attack", ascending = False)
plt.figure(figsize=(20,6))

plt.bar(df8["name"][:15], df8["sp_attack"][:15])
plt.title("Top mejores pokemon con ataques especiales")
plt.xlabel("Pokemon")
plt.ylabel("Ataque Especial / Poder")
plt.show()
```

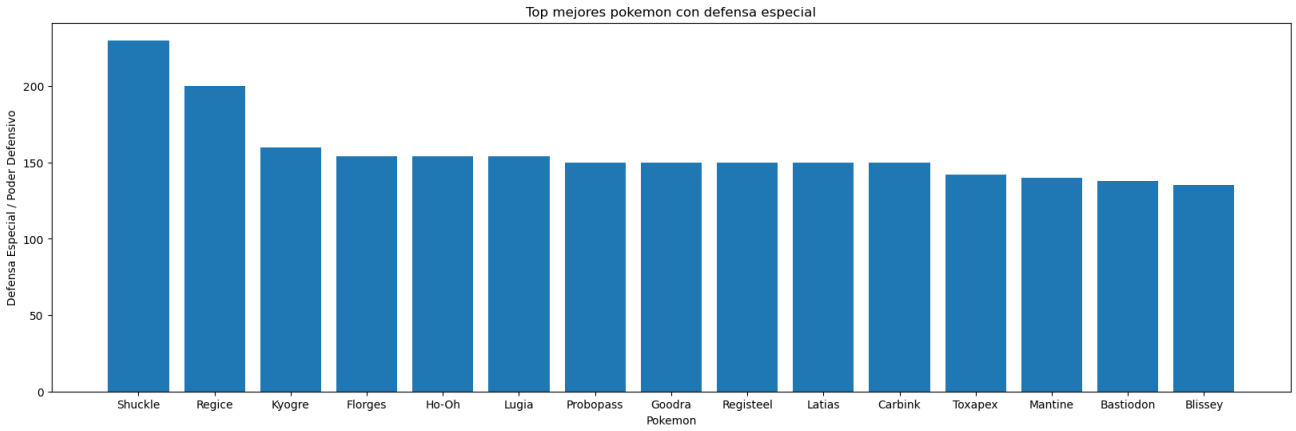


Conclusión

En el gráfico de pokemones con mejores ataques especiales, se puede concluir que los mejores pokemon son Mewtwo, Kyogre y Rayquaza. Siendo Mewtwo el pokemon con mejores stats en esta categoría (con puntos muy cercanos a los 200)

In [29]:
```
df8 = df[ ["name", "sp_defense"]].sort_values("sp_defense", ascending = False)
plt.figure(figsize=(20,6))

plt.bar(df8["name"][:15], df8["sp_defense"][:15])
plt.title("Top mejores pokemon con defensa especial")
plt.xlabel("Pokemon")
plt.ylabel("Defensa Especial / Poder Defensivo")
plt.show()
```

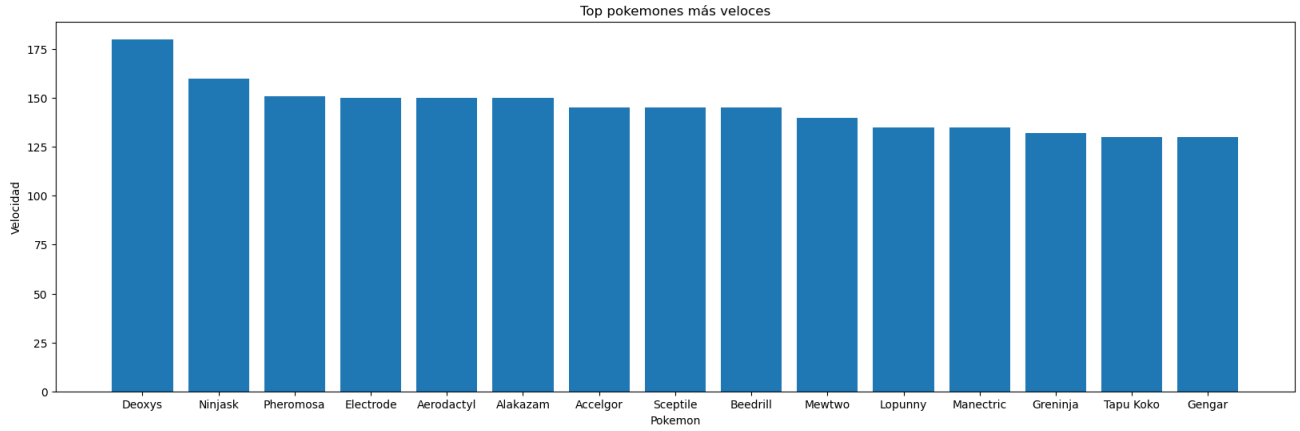


Conclusión

En el gráfico de mejores pokemones con defensa especial ("sp_defense"), se puede concluir que los mejores pokemon son Shuckle, Regice y Kyogre. Siendo Shuckle el pokemon con el mejor stat de esta categoría (alrededor de 250 puntos)

In [30]:
```
df8 = df[ ["name", "speed"]].sort_values("speed", ascending = False)
plt.figure(figsize=(20,6))

plt.bar(df8["name"][:15], df8["speed"][:15])
plt.title("Top pokemones más veloces")
plt.xlabel("Pokemon")
plt.ylabel("Velocidad")
plt.show()
```

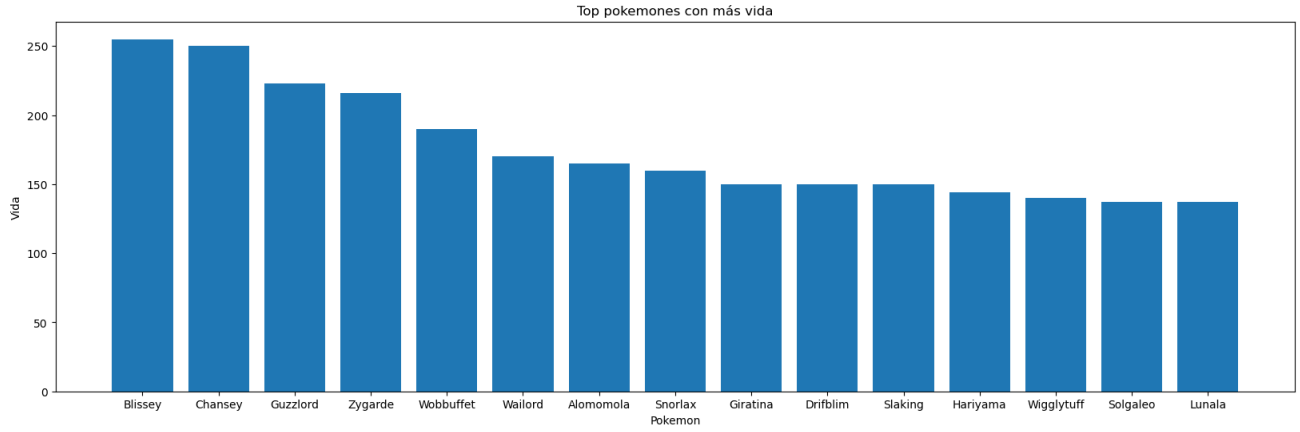


Conclusión

En el gráfico de los pokemon más veloces, se puede concluir que los mejores pokemon son Deoxys, Ninjask y Pheromosa. Siendo Deoxys el pokemon más veloz de todos (superando los 175 puntos)

In [31]:
```
df8 = df[ ["name", "hp"]].sort_values("hp", ascending = False)
plt.figure(figsize=(20,6))

plt.bar(df8["name"][:15], df8["hp"][:15])
plt.title("Top pokemones con más vida")
plt.xlabel("Pokemon")
plt.ylabel("Vida")
plt.show()
```



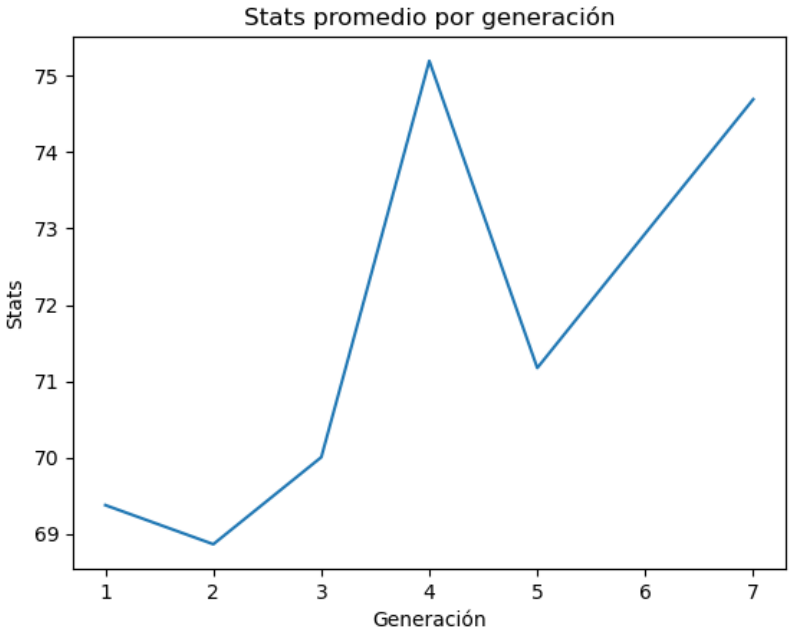
Conclusión

En el gráfico de los pokemon con más vida, se puede concluir que los mejores pokemon son Blissey, Chansey y Guzzlord. Siendo Blissey el pokemon con mayor vida de todos (superando los 250 puntos)

Ejercicio 7. ¿Cómo cambiaron los stats en promedio conforme avanzaban las generaciones?

In [53]:
```
def stats_prom(fila):
    return (fila["attack"] + fila["defense"] + fila["sp_attack"] + fila["sp_defense"] + fila["speed"] + fila["hp"]) /
df["stats_prom"] = df.apply(stats_prom, axis=1)
```

In [48]:
```
df9 = df[ ["stats_prom", "generation"]].groupby("generation").mean()
plt.title("Stats promedio por generación")
plt.xlabel("Generación")
plt.ylabel("Stats")
plt.plot( df9.index, df9["stats_prom"])
plt.show()
```



Conclusión

En este gráfico lineal, podemos observar en el eje "x" lo correspondiente a la generación de pokemons. Mientras que, en el eje "y" lo correspondiente al promedio de los stats de cada pokemon. Gracias a esta información se puede concluir que los pokemon con mejores stats fueron aquellos que pertecen o pertenecieron a la cuarta generación.

Ejercicio 8. ¿Un pokemon es más fuerte entre más difícil es de capturar?

In [49]:
```
df["capture_rate"].values
```

```
array(['45', '45', '45', '45', '45', '30', '3', '3', '3', '45', '45',
       '45', '3', '3', '45', '45', '45', '45', '45', '45', '45', '45',
       '45', '45', '255', '127', '255', '90', '255', '120', '45', '120',
       '45', '255', '120', '45', '255', '120', '45', '200', '45', '190',
       '45', '235', '120', '45', '200', '75', '255', '90', '255', '120',
       '45', '255', '120', '45', '190', '120', '45', '180', '200', '150',
       '255', '255', '60', '45', '45', '180', '90', '45', '180', '90',
       '120', '45', '200', '200', '150', '150', '150', '225', '75', '225',
       '60', '125', '60', '255', '150', '90', '255', '60', '255', '255',
       '120', '45', '190', '60', '255', '45', '90', '90', '45', '45',
       '190', '75', '205', '155', '255', '90', '45', '45', '45', '45',
       '255', '60', '45', '200', '225', '45', '190', '90', '200', '45',
       '30', '125', '190', '75', '255', '120', '45', '255', '60', '60',
       '25', '225', '45', '45', '45', '3', '3', '3', '3', '3', '3', '3',
       '3', '3', '3', '45', '3', '3', '45', '45', '45', '45', '45', '45',
       '45', '45', '45', '255', '120', '45', '255', '127', '255', '45',
       '235', '120', '45', '255', '75', '45', '45', '45', '45', '120',
       '45', '45', '120', '45', '200', '190', '75', '190', '75', '190',
       '75', '45', '125', '60', '190', '60', '45', '30', '190', '75',
       '120', '225', '60', '255', '90', '255', '145', '130', '30', '100',
```

In [50]:
```
df[ df["capture_rate"] == '30 (Meteorite)255 (Core)']["name"]
df.at[773,"capture_rate"] = 30
```

In [51]:
```
df.loc[773][["name","capture_rate"]]
```

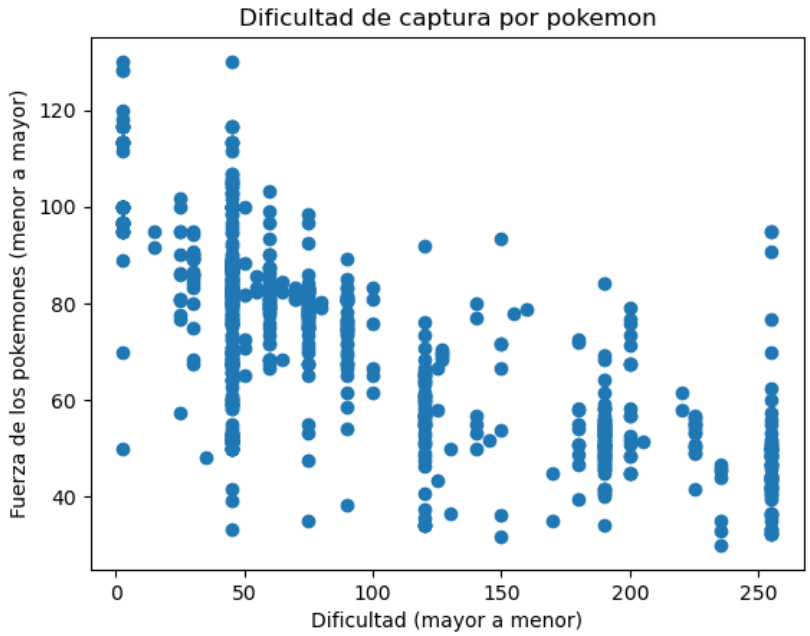
Out[51]:
```
name            Minior
capture_rate        30
Name: 773, dtype: object
```

In [52]:
```
df10 = df[ ["capture_rate", "stats_prom"]]
df10["capture_rate"] = df10["capture_rate"].astype("float")
plt.title("Dificultad de captura por pokemon")
plt.xlabel("Dificultad (mayor a menor)")
plt.ylabel("Fuerza de los pokemones (menor a mayor)")
plt.scatter(df10["capture_rate"], df10["stats_prom"])
plt.figure(figsize=(20,10))

plt.show()
```

```
C:\Users\Gabri\AppData\Local\Temp\ipykernel_23012\989391515.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df10["capture_rate"] = df10["capture_rate"].astype("float")
```




```
<Figure size 2000x1000 with 0 Axes>
```

Conclusión

El gráfico de puntos anterior muestra la fuerza de cada pokemon de forma ascendente, aundado a su dificultad para atraparlo (siendo de mayor a menor dificultad de izquierda a derecha). A pesar de solo parecer datos al azar, podemos concluir que, tentativamente (y a pesar de que hay muchos pokemon fuertes que no son tan difíciles de capturar), entre más fuerte sea un pokemon, mayor será su dificultad de captura; al menos en la mayoría de los casos.