Local Search Algorithms, hill climbing , simulated anneling, beam search, n-dueen , genetics algorithm, naive bayes, decision tree, neural network, n-gram, x-ai .. egula python code deo ar bujai dio beginer friendly please

Sure! Let's break this down step by step. Since you want **beginner-friendly explanations and Python code** for a range of AI/ML algorithms and techniques, I'll cover each one with:

1. **Short explanation**
2. **Python code example**
3. **How it works, simply**

I'll start with **Local Search Algorithms** and some others, then move to ML stuff.

---

1. Hill Climbing (Local Search)

Explanation: Hill Climbing tries to improve a solution step by step. It always moves towards a better solution, like climbing uphill to reach the highest point.

Python example (Maximize `f(x) = -x^2 + 4x`):

python Copy code

```python
import random

def f(x):
    return -x**2 + 4*x

current = random.uniform(0, 5)  # start at random x
step_size = 0.1

for i in range(100):
    neighbors = [current + step_size, current - step_size]
    next_move = max(neighbors, key=f)
    if f(next_move) <= f(current):
        break  # no improvement
```

```
    current = next_move

print("Best x:", current)
print("Max f(x):", f(current))
```

Beginner idea: Start somewhere, check small changes, go to the best change. Stop when no improvement.

---

2. Simulated Annealing

Explanation: Like Hill Climbing, but sometimes it takes worse solutions to escape local peaks. Think of **cooling metal slowly**.

Python example:

python Copy code

```python
import math, random

def f(x):
    return -x**2 + 4*x

current = random.uniform(0,5)
T = 10  # start temperature
alpha = 0.9  # cooling rate

for i in range(100):
    new = current + random.uniform(-0.5, 0.5)
    delta = f(new) - f(current)
    if delta > 0 or random.random() < math.exp(delta/T):
        current = new
    T *= alpha

print("Best x:", current)
print("Max f(x):", f(current))
```

Idea: Sometimes accept worse moves; gradually reduce randomness.

---

3. Beam Search

Explanation: Like BFS, but keeps only the **best k solutions** at each level.

Python example:

python Copy code

```python
def beam_search(start, k=2, steps=5):
    frontier = [start]
    for _ in range(steps):
        candidates = []
        for node in frontier:
            neighbors = [node+1, node-1]
            candidates.extend(neighbors)
        frontier = sorted(candidates, key=lambda x: -( -x**2 + 4*x ))[:k]
    return frontier

print("Beam search result:", beam_search(0))
```

Idea: Only keep the top `k` options instead of all possibilities.

---

4. N-Queens Problem (Backtracking)

Explanation: Place `N` queens on an `N×N` chessboard so none attacks each other.

Python example (4-Queens):

python Copy code

```python
def is_safe(board, row, col):
    for i in range(row):
        if board[i] == col or \
           abs(board[i]-col) == row-i:
            return False
    return True

def solve_nqueens(n):
    solutions = []
    board = [-1]*n
```

```
    def backtrack(row):
        if row == n:
            solutions.append(board.copy())
            return
        for col in range(n):
            if is_safe(board, row, col):
                board[row] = col
                backtrack(row+1)
                board[row] = -1
    backtrack(0)
    return solutions

print("4-Queens solutions:", solve_nqueens(4))
```

Idea: Try placing queens row by row; backtrack if conflict.

---

5. Genetic Algorithm (Optimization)

Explanation: Inspired by evolution: **selection, crossover, mutation** to find best solution.

Python example (maximize `x^2` in 0-31):

python Copy code

```
import random

def fitness(x):
    return x**2

population = [random.randint(0,31) for _ in range(6)]

for generation in range(10):
    population.sort(key=fitness, reverse=True)
    population = population[:3]  # keep top 3
    children = []
    while len(children) < 3:
        p1, p2 = random.sample(population, 2)
```

```
            child = (p1 + p2)//2
            if random.random() < 0.1:  # mutation
                child += random.choice([-1,1])
            children.append(child)
        population += children

best = max(population, key=fitness)
print("Best solution:", best)
```

Idea: Keep best, mix them, mutate a bit. Repeat.

---

6. Naive Bayes (ML Classifier)

Explanation: Probabilistic classifier using Bayes theorem. Simple and fast.

Python example:

python Copy code

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target,

model = GaussianNB()
model.fit(X_train, y_train)
print("Accuracy:", model.score(X_test, y_test))
```

Idea: Looks at each feature independently and multiplies probabilities.

---

7. Decision Tree

Explanation: Tree-like structure for decisions.

python Copy code

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target,

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
print("Accuracy:", model.score(X_test, y_test))
```

Idea: Split data based on best features recursively.

---

8. Neural Network (ANN)

Explanation: Layers of neurons that learn patterns.

python Copy code

```python
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target,

model = MLPClassifier(hidden_layer_sizes=(5,), max_iter=500)
model.fit(X_train, y_train)
print("Accuracy:", model.score(X_test, y_test))
```

Idea: Input → hidden → output; network adjusts weights to learn.

---

9. N-Gram (Text Modeling)

Explanation: Predict next word/letter using last N words/letters.

python Copy code

```python
from collections import defaultdict

text = "I love AI and I love Python"
words = text.split()
n = 2  # bigram

ngrams = defaultdict(list)
for i in range(len(words)-n):
    key = tuple(words[i:i+n-1])
    ngrams[key].append(words[i+n-1])

print(dict(ngrams))
```

Idea: Look at sequences to predict next item.

---

10. XAI (Explainable AI)

Explanation: Techniques to explain ML models (like feature importance, SHAP, LIME).

Python example with Decision Tree:

python Copy code

```python
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, export_text

data = load_iris()
model = DecisionTreeClassifier().fit(data.data, data.target)
tree_rules = export_text(model, feature_names=data.feature_names)
print(tree_rules)
```

Idea: See the “rules” the model uses to make decisions.

✅ All these examples are **beginner-friendly** and run with basic Python + `scikit-learn`.

If you want, I can make a **single Python file** containing **all of these 10 algorithms** with clear **step-by-step comments**, so you can run and experiment with everything at once.

Do you want me to do that?