Homework_2

January 21, 2024

1 Phys 41 Homework 1 Jake Anderson 1/11/2024

[1]:
```
import numpy as np
```

1.1 Problem 1: Basic numpy

[2]:
```
def wave(t, omega, n, phi):
    return np.sin((omega * t) ** n + phi)
```

[3]:
```
def quadratic(a, b, c):
    sol1 = (-1 * b + np.sqrt(b**2 - 4 * a * c + 0j)) / (2 * a)
    sol2 = (-1 * b - np.sqrt(b**2 - 4 * a * c + 0j)) / (2 * a)
    return (sol1, sol2)
```

[4]:
```
def grid_function(grid_points):
    x, y, z = grid_points
    return x * y**2 + z


x = np.linspace(-1, 1, 10)
y = np.linspace(-2, 2, 10)
z = np.linspace(-3, 3, 10)

grid_points = np.meshgrid(x, y, z)
result = grid_function(grid_points)
print(result.shape)

# For a specific point (x, y, z) on the grid, we can get the result there like␣
 ↪this:
point = [1, 2, -1]
x_index = np.where(x == point[0])[0][
    0
]  # The 2 trailing [0]'s are to get it to an integer
y_index = np.where(y == point[1])[0][0]
z_index = np.where(z == point[2])[0][0]
print(result[x_index][y_index][z_index])
```

```
(10, 10, 10)
3.0
```

1.2 Problem 2: Random numbers

[5]:
```
rng = np.random.default_rng(seed=12345)
```

[6]:
```
def coin_outcome(p):
    assert 0 <= p <= 1, print(
        "The probability `p` of getting heads must be between 0 and 1 inclusive.
 ↪"
    )
    rval = rng.random()
    if rval < p:
        return "Heads"
    else:
        return "Tails"
```

[7]:
```
def coin_outcome_array(p, n_samples):
    assert 0 <= p <= 1, print(
        "The probability `p` of getting heads must be between 0 and 1 inclusive.
 ↪"
    )
    values = [1 if (coin_outcome(p) == "Heads") else -1 for _ in␣
 ↪range(n_samples)]
    return np.array(values)
```

[8]:
```
def game(p):
    assert 0 <= p <= 1, print(
        "The probability `p` of getting heads must be between 0 and 1 inclusive.
 ↪"
    )
    counter = 0
    balance = 50
    while 0 < balance < 150:
        counter += 1
        rval = rng.random()
        if rval < p:
            balance += 1
        else:
            balance -= 1

    return counter


print(np.mean([game(0.45) for _ in range(10)]))
print(np.mean([game(0.5) for _ in range(10)]))
```

```
613.8
7398.6
```

1.3 Problem 3: More functions

[9]:
```python
def evaluate_left_to_right(tokens):
    values = []
    operators = []
    for token in tokens:
        if type(token) in [int, float]:
            values.append(token)
        else:
            operators.append(token)

    result = values[0]
    for i in range(0, len(values) - 1):
        if operators[i] == "+":
            result += values[i + 1]
        elif operators[i] == "-":
            result -= values[i + 1]
        elif operators[i] == "*":
            result *= values[i + 1]
        elif operators[i] == "/":
            result /= values[i + 1]

    return result


print(evaluate_left_to_right([2, "+", 4, "-", 5, "*", 10]))  # Should be 10
print(evaluate_left_to_right([0, "*", 35, "/", 3]))  # Should be 0
print(evaluate_left_to_right([1, "/", 2, "*", 3, "-", 4]))  # Should be -2.5
```

```
10
0.0
-2.5
```

[10]:
```python
def evaluate_properly(tokens):
    values = []
    operators = []
    for token in tokens:
        if type(token) in [int, float]:
            values.append(token)
        else:
            operators.append(token)

    # The cases that order is already left-to-right
    # Using evaluate_left_to_right() for cleanliness
    if (not ("+" in operators or "-" in operators)) or (
```

```
        not ("*" in operators or "/" in operators)
    ):
        return evaluate_left_to_right(tokens)

    # Change all [x, *, y] to [x*y, +, 0] (for / as well)
    for i in range(0, len(operators)):
        if operators[i] in ["*", "/"]:
            if operators[i] == "*":
                replacement = values[i] * values[i + 1]
            elif operators[i] == "/":
                replacement = values[i] / values[i + 1]

            operators[i] = "+"
            values[i] = replacement
            values[i + 1] = 0

    # Now that it can be done left to right, give the new list of tokens to␣
 ↪evaluate_left_to_right()
    return evaluate_left_to_right(
        [pair[i] for i in [0, 1] for pair in zip(values, operators)]
    )


print(evaluate_properly([1, "+", 3]))  # Should be 4
print(evaluate_properly([9, "*", 8]))  # Should be 72
print(evaluate_properly([2, "+", 4, "-", 5, "*", 10]))  # Should be -44
print(
    evaluate_properly([2, "+", 4, "-", 5, "*", 10, "-", 10, "*", 12])
)  # Should be -164
```

```
4
72
-44
-164
```