Homework_1

January 11, 2024

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

1.1 Problem 1: Basic python

[1]:
```
a = 5
b = 7
c = 3.14
d = 99.2
```

[2]:
```
a**c
```

[2]: 156.59064522818883

[3]:
```
((a + b) * c) + d
```

[3]: 136.88

[4]:
```
(a / b) + (c / d)
```

[4]: 0.7459389400921659

[5]:
```
list1 = list(range(0, 5))
list2 = list(range(10, 15))
list1 + list2
```

[5]: [0, 1, 2, 3, 4, 10, 11, 12, 13, 14]

[6]:
```
dict = {"a": 5, "b": 7, "c": 3.14, "d": 99.2}
print(dict["c"])
```

```
3.14
```

1.2 Problem 2: Basic functions

[7]:
```
def frequency(items, focus_item):
    assert type(items) == list, print("The `items` argument must be of type␣
 ↪`list`.")
    assert type(focus_item) in [int, float, str], print(
```

```
        "The `focus` argument must be of type `int`, `float`, or `str`."
    )

    i = 0
    for item in items:
        if item == focus_item:
            i += 1

    return i
```

[8]:
```
def sum_and_product(num_list):
    assert type(num_list) == list, print(
        "The `num_list` argument must be of type `list`."
    )
    for num in num_list:
        assert type(num) in [int, float], print(
            "The entries of `num_list` must all be of type `int` or `float`."
        )

    sum = 0
    product = 1
    for num in num_list:
        sum += num
        product *= num

    return sum, product
```

[9]:
```
def lenient_sum_and_product(num_list):
    assert type(num_list) == list, print(
        "The `num_list` argument must be of type `list`."
    )

    sum = 0
    product = 1
    for num in num_list:
        if type(num) not in [int, float]:
            print(
                f"{num} is of type {type(num)}, not `int` or `float`; ignoring␣
 ↪this entry."
            )
            continue
        sum += num
        product *= num

    return sum, product
```

1.3 Problem 3: More challenging functions

[10]:
```
# Improper function:

# def mysum(N):
#     if N<0:
#         return 1
#     else:
#         return mysum(N)+N

# Proper function:


def mysum(N):
    if N < 0:
        return 0
        """
        The problem is slightly ambiguous here, but I believe that when summing
        from 1 up to N, if N is negative the sum should evaluate to zero.
        """
    else:
        return (
            mysum(N - 1) + N
        )  # We need to perform `mysum` on the previous value, i.e. N-1


for num in [-10, 0, 1, 2, 5, 10, 100]:
    print(mysum(num))
```

```
0
0
1
3
15
55
5050
```

[11]:
```
from string import ascii_lowercase


def which_string_first(string1, string2):
    alphabet = ascii_lowercase
    min_length = min([len(string1), len(string2)])
    for i in range(0, min_length):
        if alphabet.index(string1[i]) < alphabet.index(string2[i]):
            return string1
        elif alphabet.index(string1[i]) > alphabet.index(string2[i]):
```

```
            return string2
    if len(string1) < len(string2):
        return string1
    else:
        return string2


def order(items):
    assert type(items) == list, print("The `items` argument must be of type␣
 ↪`list`.")
    for item in items:
        assert type(item) in [int, float, str], print(
            "The entries of `items` must all be of type `int`, `float`, or␣
 ↪`str`."
        )

    numbers = []
    strings = []
    for item in items:
        if type(item) in [int, float]:
            numbers.append(item)
        else:
            strings.append(item)

    # Sort the numbers
    sorted_numbers = []
    while (
        len(numbers) > 0
    ):  # We'll constantly remove the shortest from `numbers` so its length␣
 ↪will decrease
        smallest = numbers[0]
        for num in numbers:
            if num < smallest:
                smallest = num
        sorted_numbers.append(smallest)
        numbers.pop(numbers.index(smallest))

    # Sort the strings
    sorted_strings = []
    alphabet = ascii_lowercase
    while len(strings) > 0:
        first = strings[0]
        for string in strings:
            first = which_string_first(first, string)
        sorted_strings.append(first)
        strings.pop(strings.index(first))
```

```
    return sorted_numbers + sorted_strings


order([18.0, -10, 2, "era", 85, "testing", "zone", "test", 3.14])
```

[11]: [-10, 2, 3.14, 18.0, 85, 'era', 'test', 'testing', 'zone']