Programming Club Meeting 9 Slides

Functions Rvw

Basics

- Works like a function in math
- Gives lines of code a name to call that can be easily reused with different inputs
- Can also return values
- Have the ability to tell the input and output data types

Code

```
1 # Quadratic Func Example
2 # f(x) = x^2 + 3x + 7
3 def f(x):
4     print(x ** 2 + 3 * x + 7)
5
6 f(3)
7 f(6)
```

Output

```
25
61
```

Code

```
1 # Type Hinting
2 def f(x: float) -> float:
3     return x ** 2 + 3 * x + 7
4
5 y = f(3)
6 print(f"f(3) = {y}")
7 print(f"f(7) = {f(7)}")
8
```

Output

```
f(3) = 25
f(7) = 77
```

More Parameters and Defaults

Code

```
# Default Values
def f(x: float, y: float = 0) -> float:
    return x ** 2 + y ** 2 - 9

y = f(3,5)
print(f"f(3,5) = {y}")
print(f"f(7) = {f(7)}")
```

Output

```
f(3,5) = 25
f(7) = 40
```

- Add more parameters with a comma in between
- To give a default value (if none is inputted), add ‘ = val’

In and Not In

Code

```
# Is In
def main():
    print(f"+ = {isIn('+')}")
    print(f"p = {isIn('p')}")

def isIn(x):
    operations = "+-*/"
    if (x in operations):
        return True
    else:
        return False

main()
```

Output

```
+ = True
p = False
```

Code

```
# Is Not In
def main():
    print(f"+ = {isNotIn('+')}")
    print(f"p = {isNotIn('p')}")

def isNotIn(x):
    operations = "+-*/"
    return x not in operations

main()
```

Output

```
+ = False
p = True
```

Lists

Basics

- Set of stored values in a single variable
- AKA arrays and slices
- Can loop through each item in a list
- Can put lists in lists (2d array)

Code

```
# Basic List
lst = [1, 2, 3, 4, 2]
print(lst)
multi = [1, "hello", 3.8, True]
print(multi)
blankList1 = []
blankList2 = list()

print()

# Loop Through List
for num in lst:
    print(num)

print()

# 2d Array
array = [[1, 2], [2, 3], [3, 4]]
```

Output

```
[1, 2, 3, 4, 2]
[1, 'hello', 3.8, True]

1
2
3
4
2
```

Indexing a List

- Use [ ] Signs to index a list
- Can also be used the same way to index a string
- Lists and strings always start at position zero

Code

```
1 list1 = ["a", "b", "c", "d", "e", "f", "g"]
2           0    1    2    3    4    5    6
3 string1 = "abcdefg"
4            0123456
5 print("Position 2 of list: " + list1[2])
6 print("Position 1 of string: " + string1[1])
```

Output

```
c
b
```

Commands

Code

```
lst = [1, 2, 3, 4, 2]

# Indexing
print(lst[1]) # Starts at 0
lst[1] = 7
print(lst)
lst[2:5] = [9, 10]
print(lst)

print()

# Other Commands
print(f"Length: {len(lst)}")
print(10 in lst)
print(-1 in lst)
del blankList1
print(min(lst))
print(max(lst))
```

Output

```
2
[1, 7, 3, 4, 2]
[1, 7, 9, 10]

Length: 4
True
False
1
10
```

Code

```
numbers = [-1, 0, 1, 2, 3, 4, 5]
print(sum(numbers))
```

Output

```
14
```

Code

```python
# Methods
accounts = [2, 5.3, -89, 200]
accounts.append(9)
print(f"Append 9: {accounts}")
accounts.insert(1, 9)
print(f"Insert 1,9: {accounts}")
print(f"Index 9: {accounts.index(9)}")
print(f"Pop 2 return: {accounts.pop(2)}")
print(f"Pop 2: {accounts}")
print(f"Count 9: {accounts.count(9)}")
```

Output

```
Append 9: [2, 5.3, -89, 200, 9]
Insert 1,9: [2, 9, 5.3, -89, 200, 9]
Index 9: 1
Pop 2 return: 5.3
Pop 2: [2, 9, -89, 200, 9]
Count 9: 2
```

Methods

Methods Cont.

Code

```
# Methods
accounts = [2, 9, -89, 200, 9]
if (True): # Copy by reference
    accounts2 = accounts
    accounts[0] = 7
    print(f"Accounts2: {accounts2}")
accounts2 = accounts.copy()
print()
accounts[0] = 93
print(f"Accounts2 Copied: {accounts2}")
print()
accounts.sort()
print(f"Sorted: {accounts}")
accounts.extend(accounts2)
print(f"Extend: {accounts}")
accounts.reverse()
print(f"Reverse: {accounts}")
accounts.clear()
print(f"Clear: {accounts}")
```

Output

```
Accounts2: [7, 9, -89, 200, 9]

Accounts2 Copied: [7, 9, -89, 200, 9]

Sorted: [-89, 9, 9, 93, 200]
Extend: [-89, 9, 9, 93, 200, 7, 9, -89, 200, 9]
Reverse: [9, 200, -89, 9, 7, 200, 93, 9, 9, -89]
Clear: []
```

Tuples

Same as list but can’t change anything

Code

```
1  # Tuple
2  t = (3,5)
3  print(t)
4  # t[0] = 8 # causes error
5
6  # Returning w/ Tuple
7  def getLocation():
8      return (8,1)
9  x, y = getLocation()
10 print(f"x: {x}, y: {y}")
```

Output

```
(3, 5)
x: 8, y: 1
```

Practice

Practice Problem 1: Number of Triangles

Relevant Information:

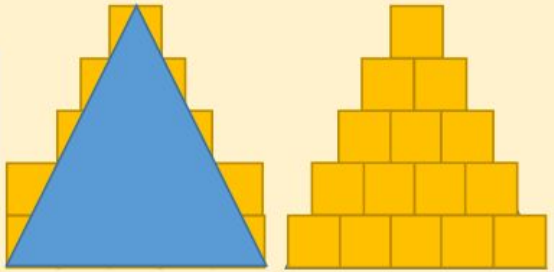
- Each layer upward has 1 less triangle
- Top layer has only 1 triangle

3 TILES COVERING TRIANGLE (SCORE: 12)

Assume that you covering a triangular-shaped area with tiles that are 10 X 10 inches. The area is an equilateral triangle. The tiles will be arranged in rows with one row aligned with one of the sides. Call this side the base. The height of an equilateral triangle is .866 times the base. The tiles are directional so that when a tile is cut to fit the slope on one edge, the remaining piece of tiles cannot be used anywhere else. The vertical edges do not have to be aligned. Given the length of the base, calculate the minimum number of tiles that are needed to completely cover the area.

Example 1:
Enter length of a side: 10
Number of tiles: 1

Example 2:
Enter length of a side: 50
Number of tiles: 15



Practice Problem 2:

The World in 2050

- Src: https://www.101computing.net/the-world-in-2050/
- Goal: Write a Python program which will determine the year the world population reaches 10 billion.
- Relevant Information:
  - Assume a growth rate of 1% per year (1.01 * pop each year)
  - Assume population in 2022 (start) is 7,850,000,000

Practice Problem 3: Count Even and Odd

- Src: https://www.geeksforgeeks.org/python-program-to-count-even-and-odd-numbers-in-a-list/
- Goal: Write a Python program that will count and output the even and odd numbers in a list. You can assume lists only include integer numbers.
- Ex:
  - List in - [2, 7, 5, 64, 14]
    - Output - Even = 3, Odd = 2
  - List in - [12, 14, 95, 3]
    - Output - Even = 2, Odd = 2