Programming Club Meeting 4 Slides

Review - Strings and Loops

Strings

- Indexing: string[2]
- Methods
- Escape characters:
  - \\
  - \t
  - \”
  - \’
  - \n
- F-strings
  - “”” Df

    Line 2”””

  - f”The number is {7}.”

Code

```python
string = "hello world!"
#         0123456789
#                  -3

# Length of String
print(len(string))

print()

# String Methods
n = string.count("l")
print(n)
i = string.find("o") # returns -1 if not found
print(i)
i = string.index("ld") # throws an error if not found
print(i)
i = string.upper()
print(i)
print(string)
i = i.lower()
print(i)
s = "  ajdsf  "
s = s.strip()
print(s)
num = "7"
num = num.zfill(3)
print(num)
```

Output

```
12

3
4
9
HELLO WORLD!
hello world!
hello world!
ajdsf
007
```

Loops

- For loop
  - Count
  - Go through every character
- While loop
  - Follows condition
- Loop keywords
  - Continue
  - Break

Code

```python
# For Loop
for i in range(7):
    print(i)
```

Output

```
0
1
2
3
4
5
6
```

Code

```python
# While Loop
string = "34"
while (len(string) < 7):
    string = "0" + string
print(string)
print("1234567")
```

Output

```
0000034
1234567
```

Chained Comparison and Nested Loops

Code

```python
# Chained comparison
num = int(input("Enter the pos of the starting letter: "))
num += 64
while (65 <= num < 91): # Checks that num is a letter
    print(f"{num-64}: {chr(num)}")
    num += 1

print()

# Use Cautiously
# Good:
print(0 < 5 <= 6)
print(5 >= 2 > 0)
print(1 == 1 == 1)

print()

# Ok:
a = 3
b = 1
c = 3
print(a != b != c)
print(a != b != c != a)

print()

# Bad:
print(0 < 5 > 3)
print(0 < 7 >= 8 != 7 == 4)
```

Output

```
23: W
24: X
25: Y
26: Z

True
True
True

True
False

True
False
```

Code

```python
# Nested Loops
for row in range(3):
    currRow = ""
    for column in range(3):
        currRow += f"{row*3+(column+1)} "
    print(currRow)
```

Output

```
1 2 3
4 5 6
7 8 9
```

Commonly used: i and j instead of row and column

What is a Function

- Block of code that can be easily called
- Give lines of code a name to call
  - Think of it like a variable for lines of code
- Like a mini-program (solves a specific problem)
  - Underscore means ignore
- Parts of a Function
- Method is a name for functions that are connected to a specific data type

Code

```python
# Basic Function
def func(x):
    print(x)
    print(x-1)
    print()

func(7)
func(12)

print()

# Solve a problem
def pow(base, exponent):
    curr = 1
    for _ in range(exponent):
        curr *= base
    print(curr)

pow(2,3)
pow(10,4)

print()

# Method
print("7".zfill(3))
```

```
7
6

12
11

8
10000

007
```

Passing Data

- Parameters - variables in the function definition
- Arguments - variables sent to the function when called (in function call)
  - What a call is
- Default values
- “Keyword arguments”

Code

```python
# Default Values
def isTrue(val=0):
    if (val):
        print("Yes")
    else:
        print("No")

isTrue(7)
isTrue()

print()

# Keyword Arguments (Setting Specific Vals)
def myFunc(needs, foo=7, bar=9):
    print(needs)
    print(foo)
    print(bar)
    print()

myFunc(2)
myFunc(5, 6)
myFunc(5, bar=6)
myFunc(5, bar=3, foo=4)
```

Output

```
Yes
No

2
7
9

5
6
9

5
7
6

5
4
3
```

Void, Returning, and Type Hinting

- Void functions don't return any values
  - Usually print something
- Non-void functions can return any data type
- Can tell Python the data types of variables in functions
  - Use the name of the data type
  - Arrow points to return type
  - If void, return type is "None"
  - Don't *have* to follow

Code

```python
# Value Returning
def pow(base, exponent): # Don't redefine in actual code
    curr = 1
    for _ in range(exponent):
        curr *= base
    return curr

pow(2,3)
print(pow(10,4))

print()

# Data Types
def plus(x: float, y: float) -> float:
    return x + y

print(plus(2.5,3.2))
print(plus("hello","world"))
```

Output

```
10000

5.7
helloworld
```

When to Type Hint

Here are a few rules of thumb on whether to add types to your project:

- If you're just beginning to learn Python, you can safely wait with type hints until you have more experience.
- Type hints add little value in short throwaway scripts.
- In libraries that will be used by others, especially ones published on PyPI, type hints add a lot of value. Other code using your libraries needs these type hints to be properly type checked itself. For examples of projects using type hints, see `cursive_re`, `black`, our own Real Python Reader, and Mypy itself.
- In bigger projects, type hints help you understand how types flow through your code, and are highly recommended, all the more so in projects where you cooperate with others.

Ex: Uppercase Function, docstring

Code

```python
# James Taddei
# Uppercase Function
# 10/19/22

def uppercase(lowercase: str) -> str:
    """
    This function turns an inputted string into the uppercase version of
    itself which is then returned.
    """
    newString = ""
    for char in lowercase:
        num = ord(char) - 32
        # Letter -> binary num, shifted 32 to be upper
        if ((num < 65) or (num > 91)):
            # not a lowercase letter, just copy
            newString += char
            continue
        newString += chr(num) # Is lowercase letter

    return newString

print(uppercase("Hello World!"))
```

Output

```
HELLO WORLD!
```

Why Use Functions?

- Can easily reuse code
- Can run code multiple times without having to rewrite even if the runs are not back to back
  - If back to back, use a loop
- Allows you to easily test subparts
- Multiple people can work together by making functions
- Easily code the program one part at a time (leads to next point)

“Think like a programmer”

- Programming isn’t about the “magic words.” You want to find a to organize the words to solve a problem
- Split the program into different segments which you solve individually
- Pass keyword to create a blank function
- Can just return single expected value for testing

Code

```
# Pass Function and Single Return
def passFunc():
    pass
def seven(x):
    return 7

passFunc()
print(seven(8))
```

Output

```
7
```

Scope

- Functions follow the Vegas Principle
- Scope can be global or specific to a function (“country”)
- “Countries” can access their own data as well as global data, but not other countries’
  - VS Code names
- Don’t want to write most things in global
  - Exception are constants
- Main function
  - Ideal main calls other functions
  - Function must be defined before use (calling main at end solves this)

Code

```python
# Scope World Example
globalPop = 8_000

def canada():
    canadaPop = 40
    print(f"Canada Pop: {canadaPop}")
    print(f"Global Pop: {globalPop}")

def america():
    pop = 330
    print(f"America Pop: {pop}")
    # print(f"Canada Pop: {canadaPop}") # causes an error

def mexico():
    pop = 130 # can reuse names, arguable if should
    print(f"Mexico Pop: {pop}")

canada()
america()
mexico()
```

Output

```
Canada Pop: 40
Global Pop: 8000
America Pop: 330
Mexico Pop: 130
```

```
Line 12: NameError: name 'canadaPop' is not defined
```

Code

```python
# Main Function
def main():
    num = float(input("Num: "))
    print(plusSeven(num))

def plusSeven(num: float) -> float:
    return num + 7

main()
```

Input Prompt

Num:

23

Output

```
30.0
```

Example Problem: isVowel Function

- Goal: Write a Python function that finds if the inputted character (string) is a vowel and returns the correct boolean.
- Include this function in a program which will test the function.
- Example tests:
  - “a”, “O”, “k”, “ae”, “8”

Practice Problem 1: Good Luck

1. Good Luck

This problem deals with numbers comprised of three decimal digits. We allow leading zeros, so numbers like 007 and 023 count. Among all 3-digit numbers, those that start and end with the same digit are *lucky*; all others are unlucky. Here is a way to transform any 3-digit number K into a lucky number.

- Let x be the number formed by writing the digits of K in ascending order.
- Let y be the number formed by writing the digits of K in descending order.
- Let z be the number consisting of the median digit of K written 3 times.
- Calculate $x + y - z$.

For example, if $K = 895$ then we calculate as follows.

- x = 589
- y = 985
- z = 888

The resulting lucky number is 686 = 589 + 985 - 888.

Write a program that prompts the user to enter a 3-digit number and then outputs the corresponding lucky number. The following execution snapshots show the required I/O format.

```
Enter a 3-digit number: 123
Good luck: 222

Enter a 3-digit number: 501
Good luck: 414

Enter a 3-digit number: 23
Good luck: 121

Enter a 3-digit number: 845
Good luck: 757

Enter a 3-digit number: 7
Good luck: 707
```

Practice Problem 2: maxNum Function

- Goal: Write a Python function that finds the max of three numbers.
- Also write a program that shows this functionality.
- Relevant Information:
  - Avoid naming variables “max” or “min” in Python

Practice Problem 3: caseCalc Function

- Goal: Write a Python function that will determine the number of upper and lowercase letters in a string.
- Also write a program that will test this functionality.
- Relevant information:
  - You can print the number of upper and lower case letters
  - Or you can use “return (lower, upper)” with the call to function being: “lower, upper = caseCalc(string)”

Practice Problem 4: isPrime Function

- Goal: Write a Python function that will determine if the inputted number is prime or not.
- Also write a program that will test this functionality.
- Relevant Information:
  - A prime number is a number which is only divisible by 1 and itself.
  - Tests:
    - Prime: 2, 13, 73, 97
    - Composite: 6, 8, 56, 63
    - Other: 1

Next Meeting:

- Review everything covered
- Try to get a better understanding
- Maybe lists

ACM Code Jam 2023

DSUACM
Student Chapter
Association for Computing Machinery

==========
>>
Computer Programming Contest for all students in grades 9-12 from districts represented by Colonial and Carbon Lehigh Intermediate Units. Teams of 4 will be given a certain number of problems to solve and the team that correctly completes the most problems will win the competition.
>>
==========
>>
>>When: December 20, 2022 (8:30 am - 11:30 am),
Snow Date: March 10, 2023
>>Where: DeSales University,
2755 Station Ave, Center Valley, PA 18034
>>How: For more information and to register your team, visit www.ciu20.org/codejam/
>>Registration Deadline: December 1, 2022
Refreshments will be provided.
Lunch will be on your own.
>>
==========
>>
>>Points of Contact:
>>
>>Heather Heimer (IU #20) hheimer@ciu20.org
>>Diahann Ouly (IU #21) oulyd@cliu.org
>>
==========

COLONIAL Intermediate Unit 20 — Colonial IU #20

Carbon Lehigh IU #21

DeSales University