Programming Club Meeting 3 Slides

Review

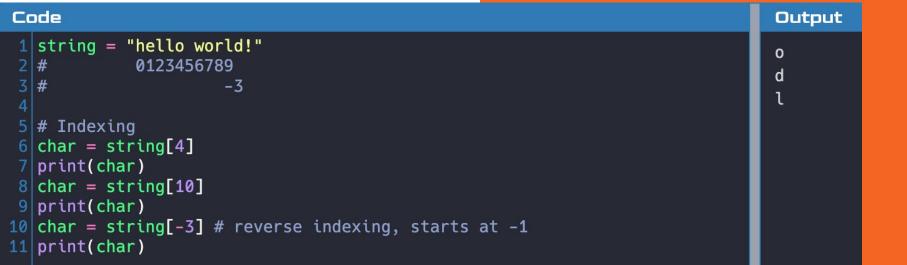
Last meeting we looked at if, else, and elif statements.

```
Code
1 \times = 12
3 # If/else
  if (True):
      print("Is true") # spacing is important in python
  if (x == 12):
      print("Is true")
10 if (x == 15):
      print("Is true") # doesn't print
      print("Is false")
  if (x == 15):
                                    Output
  elif (x == 12):
      print("12")
                                    Is true
      print("Other")
                                     Is true
  print()
                                    Is false
                                     12
```

Strings

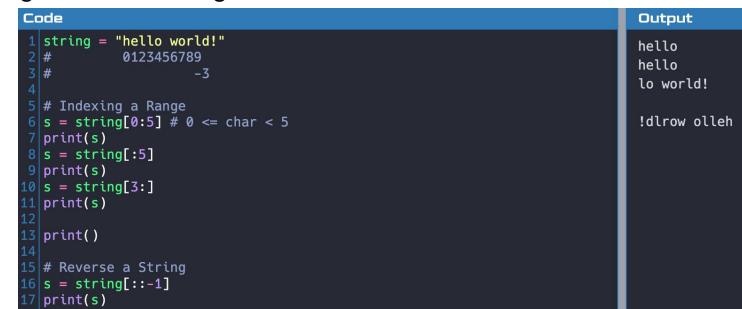
Indexing a String

- String is a series (or string) of characters mushed together into one variable
- Can get or change specific characters in a string by "indexing"



Indexing Cont.

- Can also index a range of characters
- Use indexing to reverse a string



String Methods

Strings have commands "attached" to them so that you don't have to write everything

```
Code
                                                                                Output
1 string = "hello world!"
                                                                                12
             0123456789
5 # Length of String
6 print(len(string))
                                                                                HELLO WORLD!
8 print()
                                                                                hello world!
                                                                                hello world!
  # String Methods
                                                                                ajdsf
  n = string.count(|"l")
                                                                                007
    = string.find("o") # returns -1 if not found
      string.index("ld") # throws an error if not found
    = string.upper()
  print(string)
21|print(i)
  print(s)
  num = num.zfill(3)
  print(num)
```

Methods Cont.

- endswith
- format
- islower
- isupper
- replace
- rfind
- rindex
- split & splitlines
- startswith
- Can find everything here

Escape Characters

Code	Dutput
<pre># Escape Characters print("Hello\tworld") print("Hello\\world") # print("Hello\world") # causes an error print("Hello\"world") print("Hello\'world") print('Hello\'world') print('Hello\'world') print("'Hello\\nworld\n!'")</pre>	Hello world Hello\world Hello"world Hello'world Hello'world 'Hello world !'

F-strings

Binary Number to Letter

Chr

```
Code
                                                                               Output
1 # James Taddei
2 # Number to Letter Converter
  # 10/12/22
  # Variable declaration
6 alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
8 # User input
9 num = int(input("Enter num: "))
0 num -= 65
  # Final output
  if (num < 0):
      print("Num too small")
15 elif (num > 25):
      print("Num too big")
  else: # Valid number
      print(alphabet[num])
```

Loops

Why Use Loops

- Need more complexity beyond selection (if)
- "Repetition" (iteration): repeats code within a block
- Want to have code run multiple times w/o copying it
- Might not know how many times something should run
- Want to run it indefinitely

While Loop

```
1 # While Loop
string = "34"
while (len(string) < 7):
    string = "0" + string
print(string)
print("1234567")</pre>
```

Infinite Loops

```
Code
                                                                                  Output
                                                                                  24
  string = "34"
                                                                                  34
                                                                                  34
  # Accidentally Infinite Loop
                                                                                  34
  while (string != "123"):
      print(string)
                                                                                  34
                                                                                  34
                                                                                  34
  while (True):
                                                                                  34
      print(string)
                                                                                  34
                                                                                  34
                                                                                  34
```

Repeating Calculator

Code

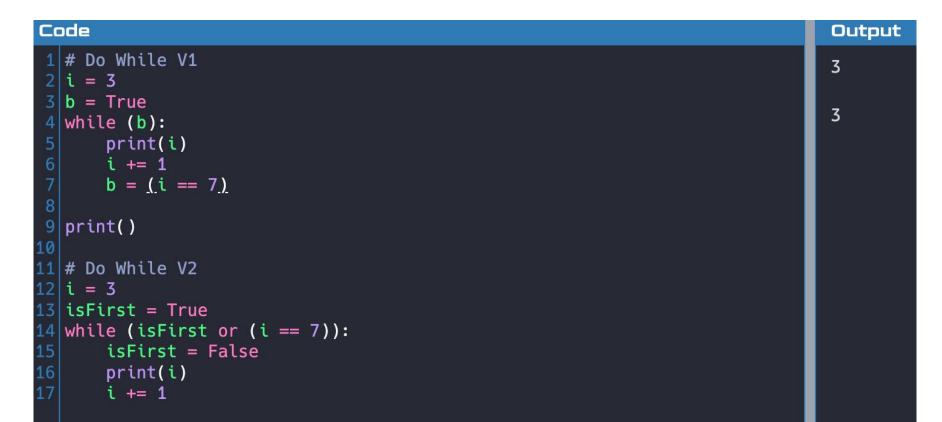
```
# James Taddei
# Repeating Calculator
# 10/12/22
# Variable declaration
operations = "+-*/"
# User input
operation = input("Enter the operation (ex: '+'): ")
while (operation in operations): # If is a valid operation
    # User input pt 2 (electric boogaloo)
    num1 = float(input("Enter num 1: "))
    num2 = float(input("Enter num 2: "))
    # Final outputs
    if (operation == '+'):
        print(num1 + num2)
    elif (operation == '-'):
        print(num1 - num2)
    elif (operation == '*'):
        print(num1 * num2)
    else:
        print(num1 / num2)
    # User input pt 3
    operation = input("Enter the operation (ex: '+'): ")
```

Counting While Loop

While Else

```
Code
                                                                              Output
    W/o Break
3 while (i < 7):
      print(i)
       i += 1
6 else:
      print("didn't break")
  # W/ Break
                                                                              didn't break
  i = 0
  while (i < 7):
      print(i)
      if (i == 3): # won't happen if breaks
          break
       i += 1
16 else:
      print("didn't break")
```

Do While



For Loop

Repeats x times or through every element in a string / list

Code

```
1 # For Loop
2 for i in range(7):
3    print(i)
4
5 print()
6
7 # Plus 2
8 for i in range(0,7,2):
9    print(i)
```

Output

Looping Through Strings

```
Code
                                                                                Output
1 # Loop Through String
2 string = "hello"
                                                                                e
  for char in string:
      print(char)
  print()
  # Loop Through String w/ Indexing
                                                                                h
  for i in range(len(string)):
                                                                                е
      print(string[i])
```

For Else

```
Code
                                                                                Output
1 string = "hello"
                                                                                h
                                                                                е
  # No break
   for i in range(len(string)):
       print(string[i])
        if (string[i] == "l"):
                                                                                0
             break
                                                                                didn't break, l not found
  else:
       print("didn't break, l not found")
                                                                                h
                                                                                е
11 print()
  # Break
  for i in range(len(string)):
       print(string[i])
       if (string[i] == "l"):
           break
18 else:
       print("didn't break, l not found")
```

For Advantages

- Faster in Python than counting while
 - With 100,000 iterations:
 - Counting while ≈ 2.468 s
 - For loop ≈ 2.103 s
- Conceptually simpler if just counting
- Auto update means infinite loops are less likely

Loop Keywords

Code

Easy to accidentally make infinite loop with continue

```
Output
# Break
for i in range(7):
    if (i == 3):
        break
    print(i)
                                                                               0
print()
# Continue
for i in range(7):
    if (i == 3):
                                                                               6
        continue
    print(i)
```

Practice Problems

Example Problem:

- Write a program that from 1-100 and outputs one of the following for each number
 - If divisible by 3 and 5, output "fizzbuzz"
 - If divisible by 3 output "fizz"
 - If divisible by 5 output "buzz"
 - Otherwise, output the number

Practice Problem 1:

- Src:
 <u>https://www.101computing.net</u>
 /the-uppercase-challenge/
- Goal: Write a program which will turn every character in a string into its uppercase version
- Relevant Information:
 - Go through every char with a loop
 - Probably easiest to make a new string to store everything
 - o chr function is recommended
 - 'ord' function does the opposite of chr
 - Lowercase starts at 97 and uppercase starts at 65

Practice Problem 2:

- Write a Python program that will take in a (positive) integer and find the sum of all of its digits without using a string.
- Relevant information:
 - Keep the number as an integer and use modulus as well as another operator

Practice Problem

3:

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

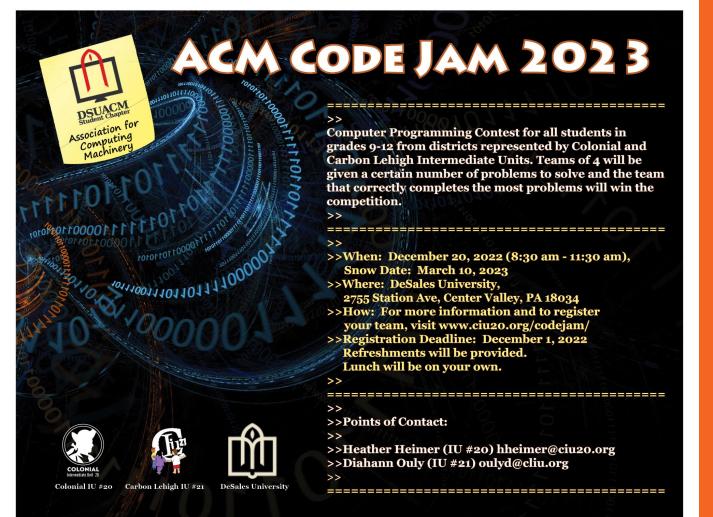
HALF CREDIT: Your program may read the input as three separate digits.

Enter a 3-digit number: 1 2 3 Good luck: 222

Enter a 3-digit number: 0 2 3
Good luck: 121

Next Meeting:

- Going to cover functions
- "Think like a programmer"



Need to know:

- Who would like to attend
- 2. How many teams
- 3. Names, members, and shirts sizes