
Programming Club

Meeting 3 Slides

Review

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

Code

```
1 x = 12
2
3 # If/else
4 if (True):
5     print("Is true") # spacing is important in python
6
7 if (x == 12):
8     print("Is true")
9
10 if (x == 15):
11     print("Is true") # doesn't print
12 else:
13     print("Is false")
14
15 if (x == 15):
16     print("15")
17 elif (x == 12):
18     print("12")
19 else:
20     print("Other")
21
22 print()
```

Output

```
Is true
Is true
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”

Code

```
1 string = "hello world!"
2 #      0123456789
3 #      -3
4
5 # Indexing
6 char = string[4]
7 print(char)
8 char = string[10]
9 print(char)
10 char = string[-3] # reverse indexing, starts at -1
11 print(char)
```

Output

```
o
d
l
```

Indexing Cont.

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

Code

```
1 string = "hello world!"
2 #      0123456789
3 #      -3
4
5 # Indexing a Range
6 s = string[0:5] # 0 <= char < 5
7 print(s)
8 s = string[:5]
9 print(s)
10 s = string[3:]
11 print(s)
12
13 print()
14
15 # Reverse a String
16 s = string[::-1]
17 print(s)
```

Output

```
hello
hello
lo world!
```

```
!dlrow olleh
```

String Methods

Strings have commands
“attached” to them so that you
don’t have to write everything

Code	Output
1 string = "hello world!"	12
2 # 0123456789	
3 # -3	3
4	4
5 # Length of String	9
6 print(len(string))	HELLO WORLD!
7	hello world!
8 print()	hello world!
9	ajdsf
10 # String Methods	007
11 n = string.count("l")	
12 print(n)	
13 i = string.find("o") # returns -1 if not found	
14 print(i)	
15 i = string.index("ld") # throws an error if not found	
16 print(i)	
17 i = string.upper()	
18 print(i)	
19 print(string)	
20 i = i.lower()	
21 print(i)	
22 s = " ajdsf "	
23 s = s.strip()	
24 print(s)	
25 num = "7"	
26 num = num.zfill(3)	
27 print(num)	

Methods Cont.

- endswith
- format
- islower
- isupper
- replace
- rfind
- rindex
- split & splitlines
- startswith
- [Can find everything here](#)

Escape Characters

Code

```
1 # Escape Characters
2 print("Hello\tworld")
3 print("Hello\\world")
4 # print("Hello"world") # causes an error
5 print("Hello\"world")
6 print("Hello'world")
7 print('Hello\'world')
8 print("'Hello\nworld\n!'")
9
```

Output

```
Hello  world
Hello\world
Hello"world
Hello'world
Hello'world
'Hello
world
!'
```


F-strings

Code

```
1 # F-strings
2 f = """'Hello
3 world
4 !'"""
5 print(f)
6 num = 7
7 f = f"Hello {num} world"
8 print(f)
```

Output

```
'Hello
world
!'
Hello 7 world
```

Binary Number to Letter

Chr

Code

```
1 # James Taddei
2 # Number to Letter Converter
3 # 10/12/22
4
5 # Variable declaration
6 alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
7
8 # User input
9 num = int(input("Enter num: "))
10 num -= 65
11
12 # Final output
13 if (num < 0):
14     print("Num too small")
15 elif (num > 25):
16     print("Num too big")
17 else: # Valid number
18     print(alphabet[num])
```

Output

C

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

Code

```
1 # While Loop
2 string = "34"
3 while (len(string) < 7):
4     string = "0" + string
5 print(string)
6 print("1234567")
7
```

Output

```
0000034
1234567
```

Infinite Loops

Code	Output
<pre>1 string = "34" 2 3 # Accidentally Infinite Loop 4 while (string != "123"): 5 print(string) 6 7 # Intentionally Infinite Loop 8 while (True): 9 print(string)</pre>	<pre>34 34 34 34 34 34 34 34 34 34 34 34 34 34 34</pre>

Repeating Calculator

Code

```
1 # James Taddei
2 # Repeating Calculator
3 # 10/12/22
4
5 # Variable declaration
6 operations = "+-*/"
7
8 # User input
9 operation = input("Enter the operation (ex: '+'): ")
10
11 while (operation in operations): # If is a valid operation
12     # User input pt 2 (electric boogaloo)
13     num1 = float(input("Enter num 1: "))
14     num2 = float(input("Enter num 2: "))
15
16     # Final outputs
17     if (operation == '+'):
18         print(num1 + num2)
19     elif (operation == '-'):
20         print(num1 - num2)
21     elif (operation == '*'):
22         print(num1 * num2)
23     else:
24         print(num1 / num2)
25
26     # User input pt 3
27     operation = input("Enter the operation (ex: '+'): ")
```

Counting While Loop

Code

```
1 # Counting While Loop
2 i = 0
3 while (i < 7):
4     print(i)
5     i += 1
```

Output

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


While Else

Code

```
1 # W/o Break
2 i = 0
3 while (i < 7):
4     print(i)
5     i += 1
6 else:
7     print("didn't break")
8
9 # W/ Break
10 i = 0
11 while (i < 7):
12     print(i)
13     if (i == 3): # won't happen if breaks
14         break
15     i += 1
16 else:
17     print("didn't break")
18
```

Output

```
0
1
2
3
4
5
6
didn't break
0
1
2
3
```

Do While

Code

```
1 # Do While V1
2 i = 3
3 b = True
4 while (b):
5     print(i)
6     i += 1
7     b = (i == 7)
8
9 print()
10
11 # Do While V2
12 i = 3
13 isFirst = True
14 while (isFirst or (i == 7)):
15     isFirst = False
16     print(i)
17     i += 1
```

Output

3

3

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

```
0
1
2
3
4
5
6

0
2
4
6
```

Looping Through Strings

Code	Output
<pre>1 # Loop Through String 2 string = "hello" 3 for char in string: 4 print(char) 5 6 print() 7 8 # Loop Through String w/ Indexing 9 for i in range(len(string)): 10 print(string[i]) 11</pre>	<pre>h e l l o h e l l o</pre>

For Else

Code

```
1 string = "hello"
2
3 # No break
4 for i in range(len(string)):
5     print(string[i])
6     # if (string[i] == "l"):
7     #     break
8 else:
9     print("didn't break, l not found")
10
11 print()
12
13 # Break
14 for i in range(len(string)):
15     print(string[i])
16     if (string[i] == "l"):
17         break
18 else:
19     print("didn't break, l not found")
20
```

Output

```
h
e
l
l
o
didn't break, l not found

h
e
l
```

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

Easy to accidentally make infinite loop with continue

Code

```
1 # Break
2 for i in range(7):
3     if (i == 3):
4         break
5     print(i)
6
7 print()
8
9 # Continue
10 for i in range(7):
11     if (i == 3):
12         continue
13     print(i)
```

Output

```
0
1
2

0
1
2
4
5
6
```

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:
<https://www.101computing.net/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
 - 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”



ACM CODE JAM 2023

>>

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

>>>Diahaan Ouly (IU #21) oulyd@cliu.org

>>



COLONIAL
Intermediate Unit 20

Colonial IU #20



Carbon Lehigh IU #21



DeSales University

Need to know:

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