

Unit 14

Loops & List Comprehensions

Asg 3 (Coding)

for Loops

```
In [ ]: # set up notebook to display multiple output in one cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

print('The notebook is set up to display multiple output in one cell.')
```

Before starting this assignment, you may want to read the articles linked below to learn about the `sep=` and `end=` optional parameters in a Python `print` statement.

print() Syntax

The full syntax of `print()` is:

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

`sep=` optional parameter

`end=` optional parameter

Refer to the articles linked below to learn about the `sep=` and `end=`, which are optional parameters in a Python `print` statement.

print() Parameters

- **objects** - object to be printed. * indicates that there may be more than one object
- **sep** - objects are separated by `sep`. **Default value:** `' '`
- **end** - `end` is printed at last
- **file** - must be an object with `write(string)` method. If omitted, `sys.stdout` will be used which prints objects on the screen.
- **flush** - If `True`, the stream is forcibly flushed. **Default value:** `False`

The `sep=` and `end=` parameters in Python `print` statement

Python print

Problem #1: Write code that uses a **for loop** to iterate through the letters of the string "Brookfield" one-by-one and for each item (letter) prints the uppercase form of the letter followed by a space.

Desired Output

B R O O K F I E L D

```
In [3]: brookfield = "Brookfield"
        for ch in brookfield:
            print(ch.upper(), end=" ")
```

B R O O K F I E L D

Problem #2: Write code that uses a **for loop** and the **range()** function to print out the first 12 nonnegative integers in a single row with a space between each nonnegative integer.

Desired Output

0 1 2 3 4 5 6 7 8 9 10 11

```
In [4]: for x in range(12):
        print(x, end=" ")
```

0 1 2 3 4 5 6 7 8 9 10 11

Practice Problem #3: Use a **for loop** that iterates over each character in the string "Matt is good at Jeopardy!" and then prints out the desired output that is specified below.

Desired Outspout

1st Part of Output M

a
t
t
i
s
g
o
o
d

a
t
J
e
o
p
a
r
d
y
!

2nd Part of Output M a t t i s g o o d a t J e o p a r d y !

Note: There is a space between each letter and two spaces between each word.

```
In [10]: str = "Matt is good at Jeopardy!"  
for ch in str:  
    print(ch)  
  
for ch in str:  
    print(ch, end=" ")
```

M
a
t
t

i
s

g
o
o
d

a
t

J
e
o
p
a
r
d
y
!

M a t t i s g o o d a t J e o p a r d y !

Practice Problem #4: Write code that incorporates the **range()** function and produces the

specified desired output.

Desired Outspu

- a. [0, 1, 2, 4, 4, 5, 6]
- b. [8, 9, 10, 11, 12]
- c. [5, 9, 13, 17, 21, 25]
- d. [5, 4, 3, 2]
- e. [31, 24, 17, 10, 3]

```
In [12]: a = []
for x in range(7):
    a.append(x)
print(a)

b = []
for x in range(8,13):
    b.append(x)
print(b)

c = []
for x in range(5, 26,4):
    c.append(x)
print(c)

d = []
for x in range(5, 1,-1):
    d.append(x)
print(d)

e = []
for x in range(31, 2, -7):
    e.append(x)
print(e)
```

```
[0, 1, 2, 3, 4, 5, 6]
[8, 9, 10, 11, 12]
[5, 9, 13, 17, 21, 25]
[5, 4, 3, 2]
[31, 24, 17, 10, 3]
```

Problem #5: Write code that uses a **for loop** and the **range()** function to iterate over the sequence 3, 10, 17, ..., 52 and then prints out "The square of {number} minus 5 is {result}." for each item in the sequence.

Desired Outspu

![q4.JPG](attachment:q4.JPG)

```
In [14]: for x in range(3, 53, 7):
        print(f"The square of {x} minus 5 is {(x**2)-5}.")
```

The square of 3 minus 5 is 4.
The square of 10 minus 5 is 95.
The square of 17 minus 5 is 284.
The square of 24 minus 5 is 571.
The square of 31 minus 5 is 956.
The square of 38 minus 5 is 1439.
The square of 45 minus 5 is 2020.
The square of 52 minus 5 is 2699.

Problem #6: Write code that uses a **for loop** and the multiplication operator, *****, to iterate over the tuple `our_tuple = ('Python', 'love', '!', 10)` to produce the output found below:

Desired Output

PythonPythonPythonPython
lovelovelove
!!!!
40

```
In [16]: our_tuple = ('Python', 'love', '!', 10)
         for x in our_tuple:
             print(x*4)
```

PythonPythonPythonPython
lovelovelove
!!!!
40

Problem #7: Write code that uses a **for loop** to iterate over the list `sports_list = ['football', 'basketball', 'baseball']` to produce the output found below:

Desired Output

FOOTBALL
BASKETBALL
BASEBALL

```
In [17]: sports_list = ['football', 'basketball', 'baseball']

         for x in sports_list:
             print(x.upper())
```

FOOTBALL
BASKETBALL
BASEBALL

The underscore character, `_`, in for loops

- Another technique with for loops is the use of the underscore character, `_`.

- We use this character quite often in variable names, but one use of this in a for loop is as a throw-away variable used to save memory. Consider the following example that prints Hello! 5 times:

In [1]: `# The underscore character, _, in for loops`

```
for _ in range(5):
    print('Hello!', end=' ')
```

Hello! Hello! Hello! Hello! Hello!

Practice Problem #8: Write code that uses the **underscore character, _** in a **for loop** to produce the output found below:

![brookfield.JPG](attachment:brookfield.JPG)

In [22]: `for _ in range(4):
 for _ in range(3):
 print("Brookfield ", end=" ")
 print("")`

```
Brookfield Brookfield Brookfield
Brookfield Brookfield Brookfield
Brookfield Brookfield Brookfield
Brookfield Brookfield Brookfield
```

Practice Problem #9: Write code that incorporates the **underscore character** in a **nested for loop** to print out the first 6 positive integers 8 different times in tabular format (see the desired output below):

Desired Output

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

In [29]: `for x in range(1,7):
 for _ in range(8):
 print(x,end=" ")
 print("")`

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

Practice Problem #10: Write code that begins with an empty list and uses the **append()**

method and **range()** function in conjunction with a **for loop** to build the list found below:

Desired Output [8, 7, 6, 5, 4, 3, 2, 1]

```
In [30]: list = []  
for x in range(8,0,-1):  
    list.append(x)  
print(list)
```

[8, 7, 6, 5, 4, 3, 2, 1]

Practice Problem #11: Write code that uses a **for loop** to add the first 15 perfect squares and then prints out the desired output specified below.

Desired Output The sum of the first 15 perfect squares is 1240.

```
In [33]: total = 0  
for x in range(16):  
    total += (x**2)  
print(total)
```

1240

Practice Problem #12: Write code that uses a **for loop** to sum the elements in the list [1, 3, 6, 10, 15, 21, 28] and then prints out the desired output specified below:

Desired Output The seventh partial sum of the given sequence is 84.

```
In [34]: list = [1,3,6,10,15,21,28]  
total = 0  
for x in list:  
    total+=x  
print(total)
```

84

Practice Problem #13: The first 10 terms of the Fibonacci sequence are found below:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Write code that uses a **for loop** to create the Fibonacci sequence and then prints out the first 15 terms of the Fibonacci sequence on one line and the on the sum of the first 15 terms of the Fibonacci sequence on the following line (refer to the desired output found below).

Desired Output

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610

The sum of the first 15 terms of the Fibonacci sequence is 1596.

```
In [30]: def fibonacci_of(n):
          if n in {0, 1}:
              return n
          return fibonacci_of(n - 1) + fibonacci_of(n - 2)

          print([fibonacci_of(n) for n in range(11)])

[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Practice Problem #14:

This problem is about **Fizz Buzz**, a programming task that is sometimes used in interviews.

(a) Use a for loop to print out the numbers 1 to 30 (b) Use a for loop to print out all the numbers 1 to 30, but leave out any number which is divisible by 3, such as 3, 6 and 9. (c) Use a for loop to print out all the numbers 1 to 30, but leave out any number which is divisible by 5, such as 5, 10 and 15. (d) Use a for loop to print out all the numbers 1 to 30, but insert the word fizz for any number that is divisible by 3, insert the word buzz for any number that is divisible by 5 and insert the word fizz buzz for any numbers that are both divisible by 3 and 5, like 15.

```
In [19]: alist = []
          for a in range(1, 31):
              alist.append(str(a))
          print(', '.join(alist))
          print("\n")

          blist = []
          for b in range(1, 31):
              if b%3 == 0:
                  continue
              else:
                  blist.append(str(b))
          print(', '.join(blist))
          print("\n")

          clist = []
          for c in range(1, 31):
              if c%5 == 0:
                  continue
              else:
                  clist.append(str(c))
          print(', '.join(clist))
          print("\n")

          dlist = []
          for d in range(1, 31):
              if d%3 == 0:
                  dlist.append("fizz")
              elif d%5 == 0:
```



```

        dlist.append("buzz")
    else:
        dlist.append(str(d))
print(', '.join(dlist))
print("\n")

```

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30

1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26, 28, 29

1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24, 26, 27, 28, 29

1, 2, fizz, 4, buzz, fizz, 7, 8, fizz, buzz, 11, fizz, 13, 14, fizz, 16, 17, fizz, 19, buzz, fizz, 22, 23, fizz, buzz, 26, fizz, 28, 29, fizz

Practice Problem #15:

Imagine you can see the future of investing and over the next four years, the interest rate of return on investments is going to be 0.02, 0.03, 0.015, 0.06. Prompt the user for an initial investment with Python's `input()` function and use the formula below to calculate how much the investment will be worth after four years. $new\ balance = old\ balance + old\ balance \times interest\ rate$ Note the first "old balance" is the person's initial investment.

```

In [25]: interest_rates = [0.02,0.03,0.015,0.06]
p = float(input("What is your initial investment"))
new_balance = 0
new_balances = []
for x in range(len(interest_rates)):
    new_balance = p + p * interest_rates[x]
    new_balances.append(new_balance)

for year in range(len(new_balances)):
    print(f"In year {year+1}, your new balance will be ${new_balances[year]}")

```

What is your initial investment4500
 In year 1, your new balance will be \$4590.0
 In year 2, your new balance will be \$4635.0
 In year 3, your new balance will be \$4567.5
 In year 4, your new balance will be \$4770.0

Practice Problem #16:

A geometric series is a series that has a common ratio between the terms. The sum of the geometric series that starts at 1/2 and has a common ratio of 1/2 approaches the value 1. The formula that shows the sum of a geometric series which approaches 1 is below.
 $1 = 1/2 + 1/4 + 1/8 + 1/16 + \dots$ Write code that uses the geometric series above to approximate the value of 1 after 10 terms are added. Use a formatted print statement to print out how far off the geometric series approximation is to 1.

```
In [28]: total = 0
         for x in range(1,11):
             total += (0.5**x)
         print(f"The geometric series is {1-total} off from 1.")
```

The geometric series is 0.0009765625 off from 1.

```
In [ ]:
```