Culture and Coding: Python Unit 1 - Introduction to Python While Loops

Programming languages are based off of these principles:

- Perform operations on numbers (both conditional and traditional)
- Split code into smaller chunks to run as needed (functions)
- Run specific code segments based on logical conditions (if/else statements)
- Repeat code a specified number of times (loops)

In this chapter we will explore Loops.

Reading Check In Question

As a reminder, we will have this from time to time to remind you reading *before* lecture

In [ ]:
```
# given the following function
def simpleWhile(start, end):
    counter = 0
    while start < end:
        counter = counter - 1
        start = start + 1
    return counter
```

In [ ]:
```
print(simpleWhile(12,12)) # question 1
```

In [ ]:
```
print(simpleWhile(1, 0)) # question 2
```

In [ ]:
```
print(simpleWhile(0, 5)) # question 3
```

While Loop Format

- The while loop says: repeat this block of code **while** the condition is true

```
while True:
    print("infinite loop, breaks computer")
```

Ideally, we want to use conditional operators to help us evaluate the condition!

In [ ]:
```
i = 0
while i < 10:
    print(i)
    i += 1  # same as i = i + 1
```

```
0
1
2
3
4
5
6
7
8
9
```

We can also compare strings, and use input!

In [ ]:

```
check = ''
while check != 'y':
    print("I want to build a snowman!")
    check = input("Do you want to build a snowman? ")[0] # grab the first character o
```

```
I want to build a snowman!
```

In class activity

You will do this activity write in the interactive slides (look for the interactive slides link in canvas)

Scroll to this section, and you can work on the code! As always, one person works the rest directs. If the person coding is directing, you shouldn't be!

Task One

Modify the following loop to

- build a string that has `10,9,8,7...0,`
- hint: you will find one of these lines of code useful
    - `ans += str(i) + ','` or
    - `ans += f"{i},"`

In [ ]:

```
i = 10
ans = "" # string to modify / append to
## student code here


print(ans)
```

```
10,9,8,7,6,5,4,3,2,1,0,
```

Task Two

Modify the code so that is instead of printing 0, it prints "blastoff"

For example: `10,9,8,7,6,5,4,3,2,1,Blastoff!`

In [ ]:

```
i = 10
ans = ""  # string to modify / append to
## student code here
```

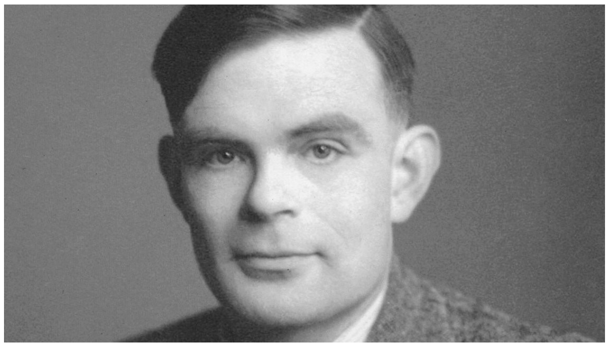
```
print(ans)
```

That is the end of loops for now! We will continue to repeat ourselves throughout the semester.

- we will have 'small' snippets of code most lectures

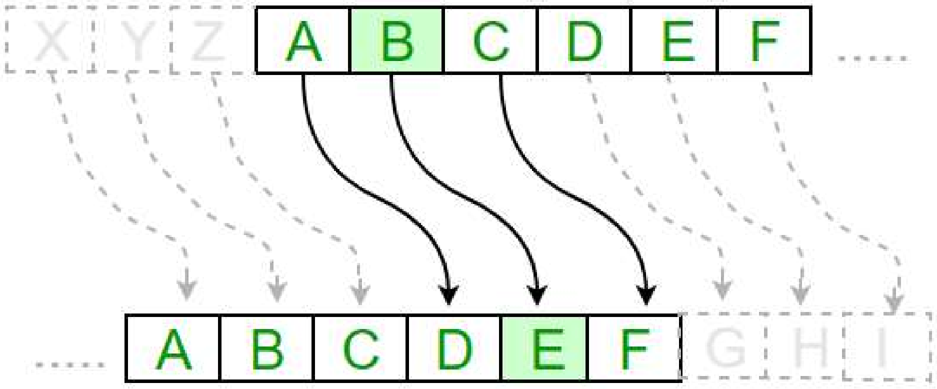
Note: with functions, loops, conditions, and operations - you can start building programs!

Alan Turing, the engima and philospher



World War II - Engima

- Built the Bombe Machine
  - Lead Programmer Jean Valentine
- Trying to break the enigma code
  - Article on writing the full machine in python
  - Based on a number of Ciphers
    - Most famously the **Caesar cipher**



- The bomb machine
  - Tested for every possible combination
  - If they found it, they found a solution
- Actually true with a lot of password breakers

    - Why you want longer passwords, more combinations
- Let's look at this in python!

```
In [ ]:
def ceasar(msg, key):
    counter = 0
    coded = ""
    while(counter < len(msg)) : # there are better loops for this, will learn about t
        coded = coded + chr(ord(msg[counter]) + key) # ord == convert to number, chr
        counter = counter + 1
    return coded

short_string = ceasar("Ada", 2)
print(short_string)
```

```
Cfc
```

- "Ada" should convert to "Cfc"
- Essentially shift the entire alphabet on over by 2 spots

```
In [ ]:
def shift_to_A(char):
    value = ord(char) # take the value and convert it to ascii
    max_shift = value - ord("A")
    if(max_shift > 25):
        max_shift = value - ord("a") # as A and a are two different characers
    return max_shift

print("TESTING", shift_to_A("C"))
print("TESTING", shift_to_A("f"))
```

```
TESTING 2
TESTING 5
```

```
In [ ]:
def max_of_group(msg):
    largest = 0
    counter = 0
    while(counter < len(msg)):
        current = shift_to_A(msg[counter])
        if current > largest:
            largest = current
        counter += 1
    return largest # returns the largest max_shift

def decode_options(msg):
    total = max_of_group(msg)
    counter = 1
    while(counter < total):
        print(ceasar(msg, -1 * counter))
        counter += 1

decode_options(short_string) # remember it is "Cfc"
```

```
Beb
Ada
@c`
?b_
```

It provided every permutation of the cipher!

- someone would have to look at it
- or have a file to compare it to known words to 'flag' certain lines
- could also shorten output by removing answers that don't use valid characters

Turing found out two characters every message had to reduce his posiblities!

Let's try something like that! Our secret messages always are a question, ending in a question mark

In [ ]:

```python
def decode_options(msg, print_option):
    total = max_of_group(msg)
    counter = 1
    while(counter < total):
        tmp = ceasar(msg, -1 * counter)
        if tmp[-1] == print_option: ## if the last character is the one we know!
            print(tmp)
        counter += 1

longer_msg = ceasar("Can Machines Think?", 2)
print("Encoded message:", longer_msg)
print("Decoded options that end with ?")
decode_options(longer_msg, '?') # we learned every message ends in a question  mark
```

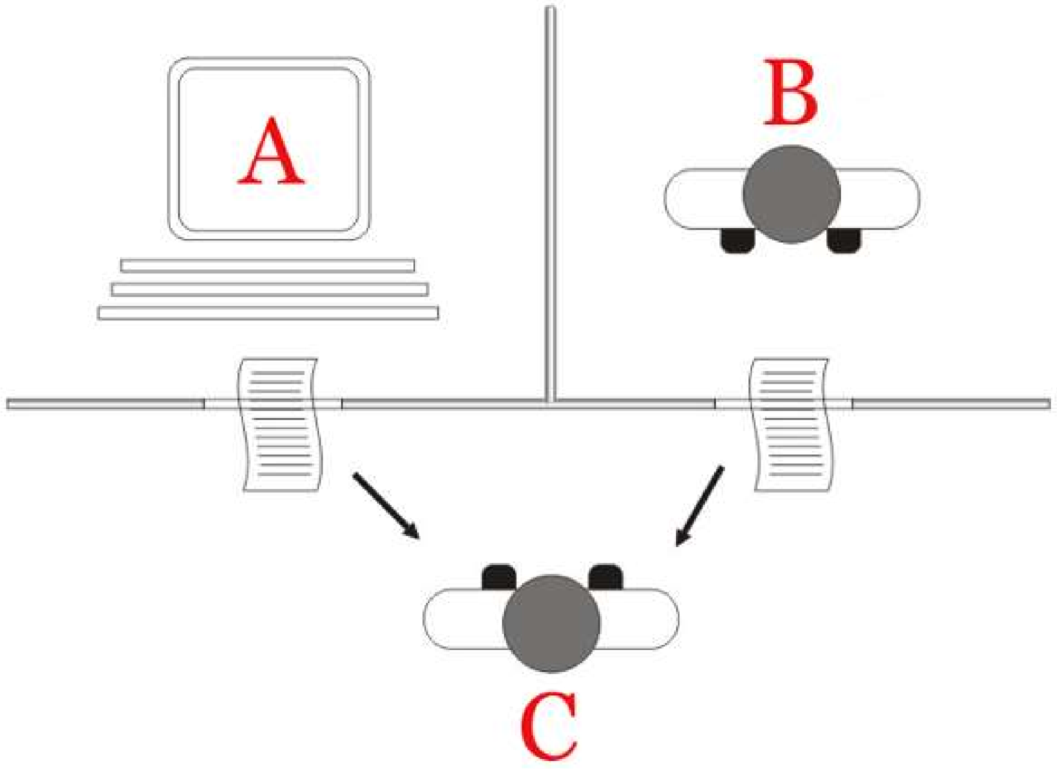
```
Encoded message: Ecp"Ocejkpgu"VjkpmA
Decoded options that end with ?
Can Machines Think?
```

Notice - only one valid option comes out!

All done with code you have already learned (except for the ord/chr command options)

The Turing Test

- Computers are estimatation of the *Turing* machine (really a von Neumann machine)
- There is also the *Turing* test
    - Intelligence and Consciousness
        - Too hard to test, not what matters
    - Can a machine make me *think* it is a human
        - That is all we really compare anyway
    - Very hard test!
        - Only considered passed just recently (million dollar prize)
        - Argumented if was actually passed



By Juan Alberto Sánchez Margallo [CC BY 2.5 (https://creativecommons.org/licenses/by/2.5)], via Wikimedia Commons

On Critique of the Turing Test

Chinese Room Argument

- John Searle
- Minds, Brains, and Programs - 1980
- Even if a computer can give proper answers:
    - Does it understand the answers?
    - Example:
        - A person is sitting in a room
        - They get a saying in a foreign language
        - The person has a 'giant book' with all possible and correct answers
        - They answer back
    - Sure the person getting the answers back - may think the person knows the other language
    - But do they?

> Discussion
>
> - Is the turing test valid?
> - What does it meant to have conciousness?
> - Can machines be intelligent?
>   - What are some issues with "trusting" machines with that much freedom?

Distinction on AI/ML

Artificial Intelligence and Machine Learning

- Techniques used to estimate search spaces and answers to problems
- Not actually saying machines can "think"

This weekend, can you see how AI/ML is affecting your life? Try to find out!