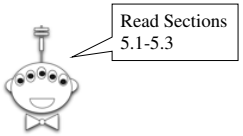
The CS 5 Black Gazette

CS 5 Penguins Fail to Return from Fall Break in Arctic

Claremont (AP): The two official CS 5 "Black" penguins failed to return from their fall-break trip to the Arctic Circle. "They went up to Northern Alaska for a little R&R before the second half of CS 5," sniffled one of the CS 5 professors. "They said they were going to just chill and maybe play a few practical jokes. I can't imagine what could have happened to them!"

Last photo taken of the CS 5 Penguins before their mysterious disappearance



Loops!

Mystery 1

I love a good mystery!

```
def leppard(argString):
    resultString = ""
    for symbol in argString:
        if symbol == "o":
            resultString = resultString + "ooo"
        else:
            resultString = resultString + symbol
    print(resultString)
```

```
>>> leppard("hello")


>>> leppard("hello to you")
```

Mystery 2

I love a good mystery!

```
vowels = ['a', 'e', 'i', 'o', 'u']

def spamify(word):
    for i in range(len(word)):
        if word[i] not in vowels:
            return word[0:i] + "spam" + word[i+1:]
    return word
```


```
>>> spamify("oui")

>>> spamify("hello")

>>> spamify("aardvark")
```

for

```
for <variable> in <iterable>:
  Do stuff!
```

```
for symbol in "blahblahblah":
```

```
for element in [1, 2, 3, 4]:
```

```
for index in range(42):
```

Three uses of for!

I'd like to see four uses of three!

Move Over XBox!

```
import random

def play():
    print("Welcome!")
    secret = random.choice(range(1, 100))
    numGuesses = 0
    userGuess = 0
    while userGuess != secret:
        userGuess = eval(input("Enter your guess: "))
        numGuesses += 1
        if userGuess > secret:
            print("Too high")
        elif userGuess < secret:
            print("Too low")
    print("You got", secret, "in", numGuesses, "guesses!")
    print("Thanks for playing!")
```

Printing strings, numbers, etc.

Try-als and Tribulations

```
def safeDivide(x, y):
    try:
        return x / y
    except:
        print("Don't DO that!")
        return float("inf")

def saferDivide(x, y):
    try:
        return x / y
    except ZeroDivisionError:
        print("Don't DO that!")
        return float("inf")
```

An Exceptional Program

```
def safestDivide(x, y):
    try:
        return x / y
    except ZeroDivisionError:
        print("Don't DO that!")
        if x == 0:
            return float("nan")
        else:
            return float("inf")
    except (ValueError, TypeError):
        print("That's just silly")
        return float("nan")
    except:
        print("I don't know what happened")
        raise
```

Input Validation

```
import random

def play3():
    print("Welcome!")
    secret = random.choice(range(1, 100))
    numGuesses = 0
    userGuess = 0
    while userGuess != secret:
        userGuess = input("Enter your guess: ")
        try:
            userGuess = int(userGuess)
        except ValueError:
            print("Please enter a number")
            continue
        numGuesses += 1
        if userGuess > secret:
# Rest of program is the same...
```

back to top

Getting `assert`ive

```
def count_to_n(n):
    assert n > 0
    for i in range(1, n + 1):
        print(i)

def fact(i):
    assert n >= 0
    if n <= 1:
        return 1
    return n * fact(n - 1)

assert fact(0) == 1
assert fact(7) == 5040
```

if n <= 0: oops ↑ raise AssertionError

Good Design

Programs must be written for people to read, and only incidentally for machines to execute. - Abelson and Sussman

1. Design your program "on paper" first. Identify the separate logical parts and the arguments and return value for each part.
2. Once your design is established, write the function "signatures" (function name, arguments) and docstrings.
3. Fill in the code for a function, **test that function carefully, and proceed only when you are convinced that the function works correctly.**
4. Use descriptive function and variable names (how about **`x, stuff, florg, jimbob`**?).
5. Don't replicate functionality. Break out repeated code into helper functions. (Often happens after the fact!)
6. Keep your code readable and use comments to help! `# Here's one now!`
7. Use whitespace liberally.
8. Avoid global variables unless absolutely necessary! Instead, pass each function just what it needs.
9. Use recursion, list comprehension, and functional constructs (e.g. map, reduce, filter, lambda) where appropriate.

An Example...

Objective: Write a tic-tac-toe program that lets two human players play, and stops when a player has won.

Functions:

main(): Welcomes user, plays a game, asks if we want to play again
welcome(): Prints the welcome message
playGame(): Maintains a board and plays one game
getMove(board, player): Queries the player (1 or 2) for their move and changes the board accordingly
printBoard(board): Takes a board as argument and displays it
gameOver(board): Evaluates a board to see if game over

```
# Tic-tac-toe by Ran Libeskind-Hadas
debug = True

def main():
    """Play tic-tac-toe with a human"""
    welcome()
    while True:
        if debug: print("About to enter playGame")
        playGame()
        response = input("Would you like to play again? (y or n): ")
        if response not in ["y", "Y", "yes", "Yes", "Yup", "si", "oui", "youbetcha"]:
            print("Bye")
            return

def welcome():
    """Prints the welcome message for the game.
       We might also print the rules for the game and any other
       information that the user might need to know."""
    print("Welcome to tic-tac-toe!")

def playGame():
    """Play one game of tic-tac-toe"""
    if debug:  print("Entering the playGame function")
    board = [[" ", " ", " "], [" ", " ", " "], [" ", " ", " "]]
    player = 1
    while not gameOver(board):
        print("The board looks like this:")
        printBoard(board)
        getMove(board, player)
        if player == 1:                    # Can this be done with clever arithmetic?
            player = 2
        else:
            player = 1
```

```
# Tic-tac-toe by Ran Libeskind-Hadas
debug = True

def main():
    """Play tic-tac-toe with a human"""
    welcome()
    while True:
        if debug: print("About to enter playGame")
        playGame()
        response = input("Would you like to play again? (y or n): ")
        if response not in ["y", "Y", "yes", "Yes", "Yup", "si", "oui", "youbetcha"]:
            print("Bye")
            return

def welcome():
    """Prints the welcome message for the game.
       We might also print the rules for the game and any other
       information that the user might need to know."""
    print("Welcome to tic-tac-toe!")

def playGame():
    """Play one game of tic-tac-toe"""
    if debug:  print("Entering the playGame function")
    board = [[" ", " ", " "], [" ", " ", " "], [" ", " ", " "]]
    player = 1
    while not gameOver(board):
        print("The board looks like this:")
        printBoard(board)
        getMove(board, player)
        if player == 1:
            player = 2
        else:
            player = 1
```

What's this?!

How 'bout:

```
row = [" ", " ", " "]
board = [row, row, row]
```

Or

```
board = [[" ", " ", " "]] * 3
```

```
def gameOver(board):
    """Returns False if the game is NOT over.  Otherwise, prints a message
       indicating which player has won and then returns True indicating that the
       game is over.  THIS FUNCTION IS NOT IMPLEMENTED CORRECTLY!"""
    return False

def getMove(board, player):
    """Takes the board and the current player (1 or 2) as arguments.
       Asks the player for a move.  If it's legitimate (position
       exists and is empty), updates the board.  Otherwise, the player
       is queried again until a valid move is provided."""
    print("Player " + str(player) + "'s turn")
    while True:
```

Fill these in!

```
def printBoard(board):
```

```
print()   # new line!
```

```
>>> board = [ ["1", "2", " "], [" ", "1", "2"], [" ", " ", "1"]]
>>> printBoard(board)
1 | 2 |
-----------
  | 1 | 2
-----------
  |   | 1
>>>
```

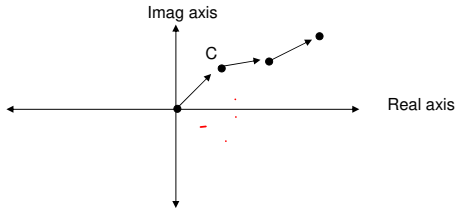
Coming Lab Problem: The Mandelbrot Set

Consider some complex number C

$z_0 = 0$

$z_{n+1} = z_n^2 + C$

For which values of C does this *not* diverge?



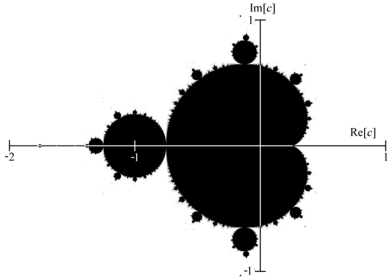
Lab Problem: The Mandelbrot Set

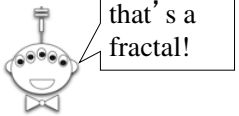
Consider some complex number C

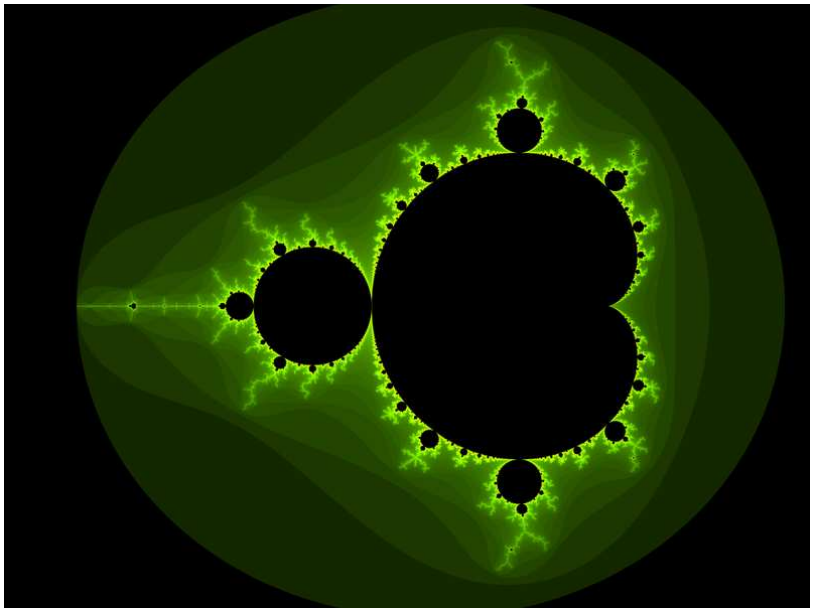
$z_0 = 0$

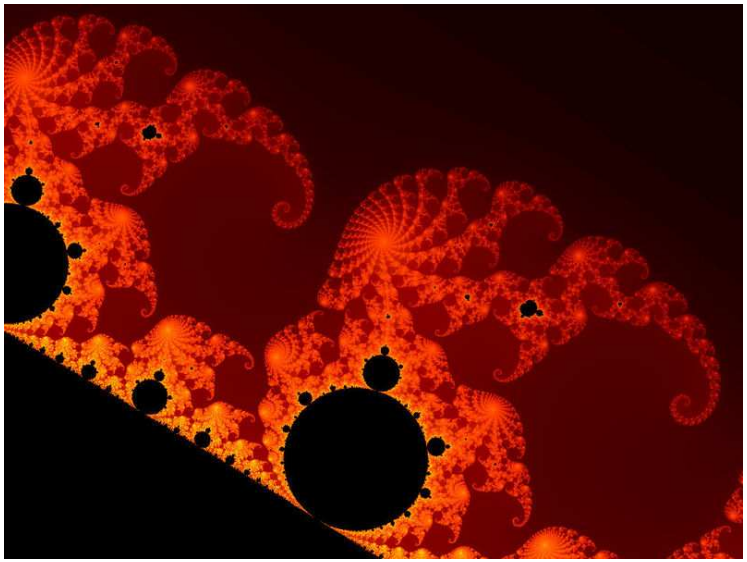
$z_{n+1} = z_n^2 + C$

For which values of C does this *not* diverge?









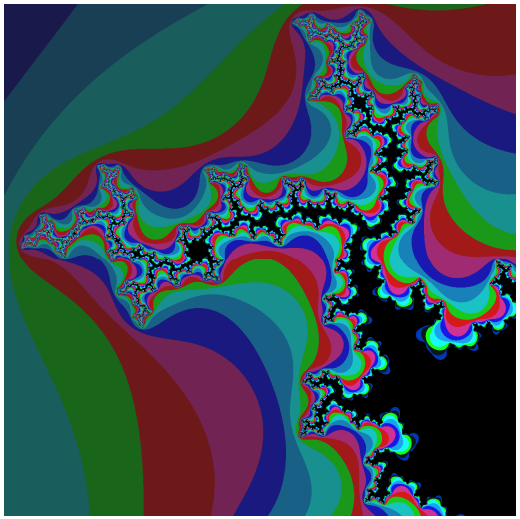


Image courtesy of Aaron Gable, CS 5 Black

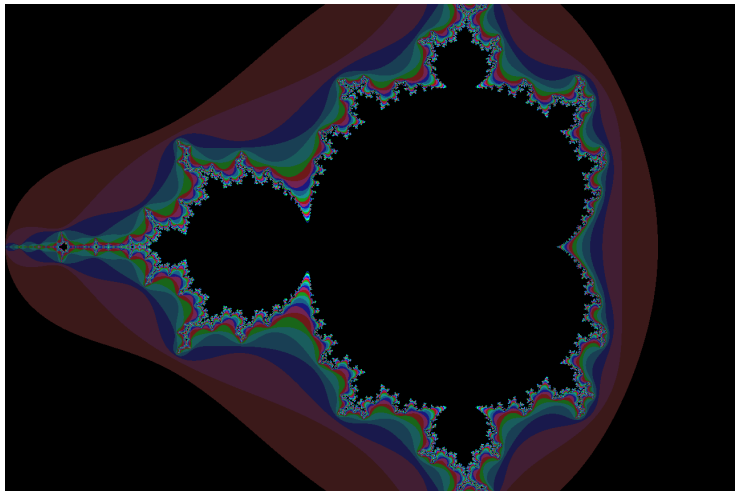


Image courtesy of Aaron Gable, CS 5 Black