Designing a Program to Solve Your Problem (1/2)
Code Examples HW 2-2
Your HW Assignment

• Write a program that accepts a single user input denoting their choice of rock, paper or scissors

• Decide whether their choice beats the computer’s random selection

• Print out the result
Control Statements

• Assume our program chose paper, we can vary the output of the program based on the user’s input with control statements

```python
users_choice = input("Rock, paper, scissors? ")

# How can we vary the output of this program
# based on user’s choice?
```

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>rock</td>
<td>lose</td>
</tr>
<tr>
<td>paper</td>
<td>draw</td>
</tr>
<tr>
<td>scissors</td>
<td>win</td>
</tr>
</tbody>
</table>
Control Statements

- Control statements *control* the flow of code
- Execution of code will depend on decision made by the control statements
- Indented lines following the control statement will only be executed if the control statement is met
- Indent lines only a single indent beyond the above control statement
- The end of each control statement must include a colon (:)
if

• if statements evaluate if a condition is True
• If so, the code indented following the if is executed
• Useful if your program must handle a single case in a special manner
• Execution resumes at the next non-indented line regardless
• Remember the colon (:)!!!!

```python
if condition :
    # Code executed if the if statement is True
    
# Code resumes normal operation
```
if examples

a = 5
b = 3
if a > b:
    print("A is greater than B")
if a < b:
    print("A is less than B")

word = "awesome"
sentence = "the movie was awesome"
if word in sentence:
    print("I’m glad you enjoyed it!")

$ python3 program.py
    A is greater than B

$ python3 program.py
    I’m glad you enjoyed the movie
if else

- When your program must handle execution of a single case and its exact opposite

- Code will be executed under either the if or the else statement. At least one, but not both

- Remember the colons!

```python
age = float(input("What is your age? "))
if age < 30:
    print("You are cool!")
else:
    print("You are not cool")
```
users_choice = input("Rock, paper, scissors? ")

if users_choice == "rock" or 
    users_choice == "scissors" or users_choice == "paper":
    print("Your input was valid")
else:
    print("Your input was not valid")
if elif else

• Enables the handling of an arbitrary number of cases

• elif is an abbreviation of else if

• For statements with more than one elif, structure code so that the else block handles the unexpected or invalid cases

```python
if condition1:
    # Some code for condition 1
elif condition2:
    # Some code for condition 2
elif condition3:
    # etc
else:
    # Case
```
Use else for invalid case

users_choice = input("Rock, paper, scissors? ")

if users_choice == "rock":
    print("You lose!")
elif users_choice == "scissors":
    print("You win")
else:
    print("It’s maybe a draw?")
users_choice = input("Rock, paper, scissors? ")

if users_choice == "rock":
    print("You lose!")
elif users_choice == "scissors":
    print("You win")
elif users_choice == "paper":
    print("It’s a draw")
else:
    print("Learn to spell")
Variable Definitions with Control Statements

- If a variable is defined within your if statement, its existence depends on the input of the program
- This will make your code bug prone!
- Best practice is to define the variable before the if statement with a default value
- Also acceptable to include the variable definition in every branch
users_choice = input("Rock, paper, scissors? ")

if users_choice == "rock" or \
    users_choice == "scissors" or users_choice == "paper":
    input_is_valid = True

if input_is_valid:
    print("Way to go!")

Code expects variable to be defined

But if user's choice was not one of the three, the variable is not defined
Variable Definitions with Control Statements

```python
users_choice = input("Rock, paper, scissors? ")

input_is_valid = False
if users_choice == "rock" or \\
    users_choice == "scissors" or users_choice == "paper":
    input_is_valid = True

if input_is_valid:
    print("Way to go!")
```

Define with a default value before if statement
Variable Definitions with Control Statements

```python
users_choice = input("Rock, paper, scissors? ")

if users_choice == "rock" or \
    users_choice == "scissors" or users_choice == "paper":
    input_is_valid = True
else:
    input_is_valid = False

if input_is_valid:
    print("Way to go!")
```

Add variable definition to all branches of if statements

Add variable definition to all branches of if statements

This is less desirable because it requires writers and maintainers to keep track of all instances of the variable’s definition
Variables inside of control statements

• If you need a variable just for the duration of the single control statement, define it inside the if statement

```python
if not input_is_valid:
    new_choice = input("Come again?")

    if new_choice == "rock" or \
       new_choice == "scissors" or new_choice == "paper":
        input_is_valid = True
        users_choice = new_choice

    # Do something with users_choice down here
```
Designing a Program
Programs Before Today

Input → Step 1 → Step 2 → Step 3 → Output
Todays Homework

Step 1

Step 2a

Step 2b

Step 3

Step 2c

Input → Step 1 → Step 2b → Step 3 → Output
Realistic Programs

Input → Clean Input

Step 2 → Step 4

Step 3a → Step 3b

Steps 5 - ? → Format Output

Output
Outlining a Program

• For ongoing assignments, we will ask you to outline your program before you write a single line of code

• This process will help you program better by understanding the structure of your program before you write any code

• You already did this in your proposal by outlining your methodology
Determine input/output
Determine Input and Output

User's freeform text input → Program

"Win", "Lose", "Draw"
Invalid Input
Create Examples of the Program’s Execution

Computer: “paper”

“rock” → Program → “Lose”
Create Examples

Computer: “paper”

“Scissors”  Program  “Win”
Create Examples

Computer: N/A

"Win" → Program → "Invalid Input"
Identify Key Points in Program

Input: freeform text

Computer’s choice: rock, paper or scissors

User’s choice: rock, paper, scissors or invalid?

Winner: user or computer?

Identify Functionality Blocks and Connect Inputs and Outputs

Input: freeform text

Computer’s choice?

User’s choice?

Winner: user or computer?

Determine Computer’s Choice
Input: None
Output: Computer’s Choice

Determine User’s Choice
Input: Freeform text
Output: User’s Choice

Decide Winner
Input: User’s & Computer’s Choice
Output: Winner

Format Output
Input: Winner

Output: “Win/Lose/Draw”
“Invalid Input”
Write Test Cases

• Designing a good program requires testing the program for correctness

• To test a program, you generate sample input and output pairs called test cases

• The examples you created before designing your program can serve as a starting point for test cases

• Extreme input examples that stress your program are called corner cases. Include many corner cases

• Testing a program that produces output based on a randomized input is difficult because the result always changes

• To test our randomized program, you can temporarily hard code the computer’s choice
Test Cases

Computer: “paper”

“rock” -> Program -> “Lose”
Test Cases

Computer: “scissors”

“Rock” — Program — “Win”
Test Cases

Program

Computer: “rock”

“ROCK” → “Draw”
Test Cases

Computer: “rock”

“Paper!” → Program → “Win”
Test Cases

Computer: N/A

Program

“I win” ——> “Invalid Input”
Test Cases

Computer: N/A

“rock scissors” → Program → “Invalid Input”