ECS 15

if and random
Testing user input using **if** statements
Truth and falsehood in Python
Getting random numbers
Flows

Sequential

- Block of instructions 1
- Block of instructions 2
- Block of instructions 3

Conditional

- Condition
- True: instructions 1
- False: instructions 2

Repeating

- For each item
- Block of instructions
Context – a quiz

- Last time: getting answers from the user with the `input()` function
- **Don’t** start by writing out the whole program
- Start by trying `input()` in IDLE
if Syntax

- if **condition**: command
- Important features: `==`, `:`, indentation
- Using the **if** statement:

```python
if answer == "yes":
    print ("Dude!")
```
If ... else...

```python
if reply == "a":
    points = 1
else:
    points = 0
```

Note else is NOT indented. Use backspace key to get back to left side.
The if statement dissected

- Look at its separate parts:

  ```python
  if answer == "yes" :
    print ("Dude!")
  ```

- if is a Python command
- print is a Python function
- "yes" and "Dude!" are strings
- Condition - answer == "yes" - is actually an expression – a new kind...
Some malformed if statements

- if answer == yes:
  print ("Dude!")

- if answer = "yes":
  print ("Dude!")

- if answer == "yes"
  print ("Dude!")

- if answer == "yes":
  print ("Dude!")

yes is a variable, should be string.

Uses = instead of ==

Missing :

No indentation
Boolean expressions

- Try it in IDLE, see what its value is
- Values of Boolean expressions are either True or False
- True and False are both built-in constant, assignments to them are illegal.
- So we have seen three kinds of expressions, numeric, string, and Boolean
In the if statement...

- if True:
  print( "This always gets printed")

- if False:
  print ("This never gets printed")

- if reply == "b":
  print ("This gets printed only if reply is 'b' ")

- if reply != "b":
  print ("This gets printed only if reply is NOT 'b' ")
Comparison operators

- ==
- !=
- <
- >
- <=
- >=
A detour on Boolean

- See next notes for next lecture.
Getting random numbers

Here is a program that picks a number between 1 and 10:

```python
# get access to random number function
import random
# use function that picks a random integer
x = random.randrange(1,10)
y = random.randrange(10)
```
Our first module

- A **module** is a collection of Python functions, maybe other stuff thrown in (library in other computer languages)
- We have to ask for the module before we can use its functions
- We ask for the module using `import`
- All the functions in module **random** are named `random.something()`
- We’ll use some modules
A program to flip a coin

- Let the user call heads or tails
- Use random to pick either 1 or 2
- If 1, it’s heads, if 2, it’s tails
Start simple, then...

- This program has **nested** if ... else... statements.
if...elif...else

if x==1:
    # only done when x==1
    print (x)
elif x<3:
    # only done when x != 1 and x<3
elif x!=5:
    # only done when x!=1 and x>=3 and x!=5
else:
    # done when x == 5 (the only case left)
More than two cases...

- Need to do something if rock, something else if scissors, and something else if paper..
- if...elif...else
- if...elif...elif...elif...else
Blocks

- A block in Python is a sequence of lines, all starting on or to the right of a particular column:

```
print( "Here is your score:")
if score == 6:
    print ("Perfect score of",end=' ')  
print (score)
```

- Blocks can be nested inside of other blocks
if not (user == "r" or user == "s" or user == "p"):
    print ("Not an allowed choice!"")
else:
    # Choose a random number between 1 and 2
    number = random.randrange(1,4)
    # 1: rock, 2: scissor, and 3: paper
    # Random number determines the program's choice
    if number == 1:
        program = "r"  # 1 means rock
    elif number == 2:
        program = "s"  # 2 means scissors
    else:  # number == 3:
        program = "p"  # 3 means paper
Rock, scissors, paper

- Play the game with your neighbor.
More than two cases...

- Need to do something if rock, something else if scissors, and something else if paper..
- if...elif...else
- if...elif...elif...elif...else
Complicated logic

- How to decide who won rock, paper, scissors?
- Make a written plan describing all cases
  My notation: user == “r”, computer == “p”, computer wins, etc.
  Write this as RP-c
  PR-u would be user == “p”, computer == “r”
  SR-c would be...?
Easy approach

- Use a separate `if..` for every possibility
  Example: see rsp1.py
- Use your plan to make sure you cover all cases!
More slick…rsp2.py

- Ties are easy to recognize
- if user == computer:
  print ("tie.")
- Handle remaining cases in pairs
  - by user’s choice, or
  - by computer’s choice, or
  - by pair of symbols
- Make sure you handle all cases!