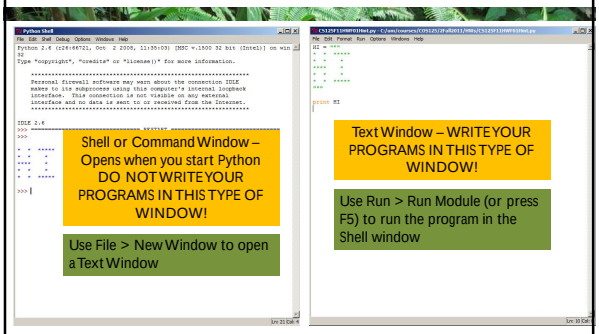
Introduction to Python Notes #2

Curtis Meadow
School of Computing and Information Technology
University of Maine

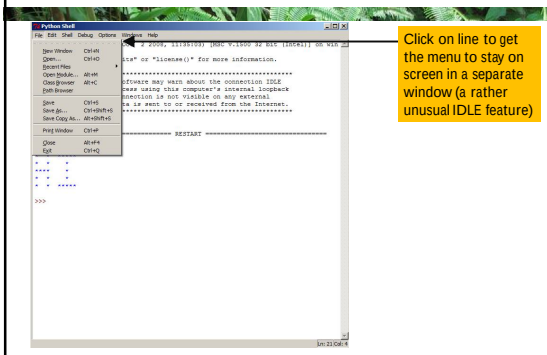
Running Python

- There are a number of different ways to run the Python interpreter
- We will use the IDLE Integrated Development Environment (IDE) almost exclusively.

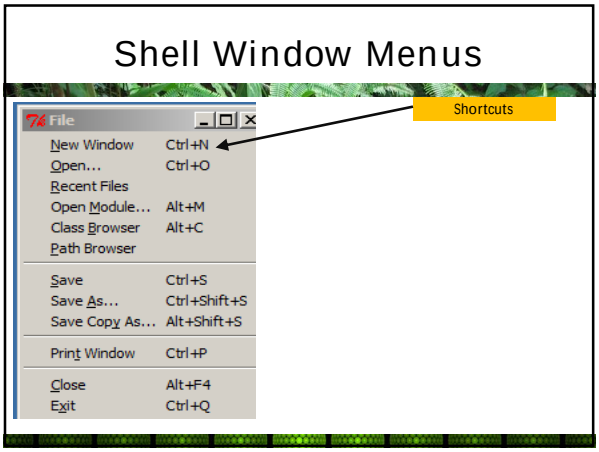
IDLE's TWO WINDOWS



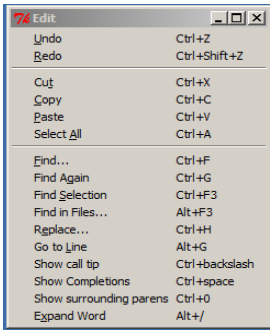
Shell Window Menus



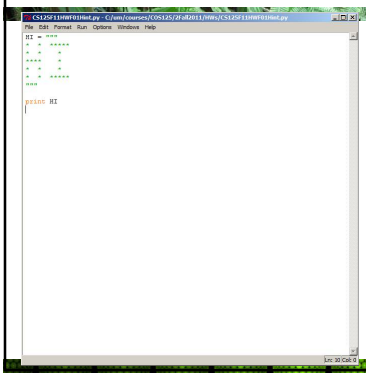
Shell Window Menus



Shell Window Menus



Text Window Menus



The File, Edit, Windows and Help Menus are the same for both Windows.

You Can Use Python Without Using IDLE

- On Unix systems and Apples, some version of Python is installed
  - Note the preinstalled version is 2.x, not 3.x
- It is typically installed without IDLE
- You can run it from a terminal window or shell window

The Command Shell in Windows

- Click on Start in the lower left corner of your main screen
- A text entry box appears at the bottom of the pop-up menu
- Enter **cmd** and hit Enter and you will get a shell
- Close command shells with the command **exit** (or use the close button in upper RH corner)

Starting Python

- You might have to have to issue a CD (change directory) command before starting Python:
  cd \python27
- Somewhat easier is to use
  Start > All Programs > Python > Python (command line)

Closing Python

- Issue the command (note the parentheses)
  quit()
- or
  exit()
- Depending on the operating system you can also use ^D (Ctrl-D) or ^Z (Ctrl-Z) to exit Python

Interrupting a Program

- It's easy to create an infinite loop

```
>>> i = 0
>>> while i < 10:
...     i = i - 1
...     print (i)
```

- Use ^C (Ctrl-C) to interrupt Python

Non-Linear Programs

- So far we have written linear programs that just have the structure

```
statement 1
statement 2
statement 3
…
statement n
```

- One exception was the house example when we used **`for`** to create a **loop**

Selection and Iteration

- Selection means testing a condition and executing different blocks of code depending on the outcome of the test
  - Often called "branching"
- Iteration means executing a block of code repeatedly
  - Often called "looping"
  - Loops are usually terminated by some conditional test

Testing Conditions

- Python syntax has three possible forms

```
if condition:
    line  1
    line  2
    …
```

```
if condition:
    line  1
    line  2
    …
else:
    line  1
    line  2
    …
```

```
if condition 1:
    line  1
    line  2
    …
elif condition 2:
    line  1
    line  2
    …
elif condition 3:
    line  1
    line  2
    …
else:
    line  1
    line  2
    …
```

Indentation

- From the Python manual:
  - "Leading whitespace (spaces and tabs) at the beginning of a logical line is used to compute the indentation level of the line, which in turn is used to determine the grouping of statements."
- In the abstract examples above indentation determines where the "then" and "else" blocks begin and end
- An unusual term used often in Python context is "dedent" (opposite of indent); more common in English is the term "outdent"

How to Shoot Yourself in the Foot #1

- Mix tabs and spaces for indents!
- Use different numbers of spaces for indent leves!
- The recommended Python standard is spaces only, NO tabs, and 4 spaces per indent level
- Some editors, such as Notepad++, will display whitespace as symbols.

An if Statement

- What will the output look like if 5 is entered in response to the prompt for input in the following program?

```
val = input('Please enter 1, 2 or 3. ')
if val == '1':
    print ('1 was entered')
elif val == '2':
    print ('2 was entered')
elif val == '3':
    print ('3 was entered')
else:
    print ('Something other than 1, 2 or 3 was entered')
```

Answer

```
>>>
Please enter 1, 2 or 3. 5
Something other than 1, 2 or 3 was entered
>>>
```

Question

- What will the output look like if 2 is entered in response to the prompt for input in the following program?

```
val = input('Please enter 1, 2 or 3. ')
if val == 1:
    print ('1 was entered')
elif val == 2:
    print ('2 was entered')
elif val == 3:
    print ('3 was entered')
else:
    print ('Something other than 1, 2 or 3 was entered')
```

Answer

```
>>>
Please enter 1, 2 or 3. 2
Something other than 1, 2 or 3 was entered
>>>
```

WHY?

Strings, Integers and Floats

- A brief digression (more later)
- Internal Representation in bits:

2	00000000 00000000 0000000 00000010
'2'	00110010
2.0	01000000 00000000 0000000 00000000
-2	11111111 11111111 11111111 1111110
'-2'	00101101 00110010
-2.0	11000000 00000000 0000000 00000000
'0.2'	00111000 00101110 00110010
0.2	00111110 10011001 10011001 1001101

(Don't panic! You don't need to know this material to learn Python – it's just to illustrate a point)

Fixed!

- Because 2 is not the same as '2' we change comparisons to use strings

```
val = input('Please enter 1, 2 or 3. ')
if val == '1':
    print ('1 was entered')
elif val == '2':
    print ('2 was entered')
elif val == '3':
    print ('3 was entered')
else:
    print ('Something other than 1, 2 or 3 was entered')
```

Comparison?

- **A == B # is A equal to B?**
  - if A = B is a syntax error
- **A > B**
- **A < B**
- **A <= B**
- **A >= B**
- **A != B # is A not equal to B?**
- These six operators are called the "*relational operators*"

More Complex Conditions

- Conditional expressions can be combined with **AND** and **OR**; negated with **NOT**
- **`Condition1 and Condition2`**
- **`Condition1 or Condition2`**
- **`not Condition`**
- **`Condition1 and not Condition2`**
- etc….

Boolean Values

- Boolean values are the results of comparisons
- Only two Boolean values exist:
  `True`
  `False`

Boolean Operators

- 5.2. Boolean Operations — and, or, not
- These are the Boolean operations, in ascending priority:

Operation	Result
x or y	if *x* is false, then *y*, else *x*
x and y	if *x* is false, then *x*, else *y*
not x	if *x* is false, then **True**, else **False**

- not has a lower priority than non-Boolean operators, so not a == b is interpreted as not (a == b), and a == not b is a syntax error.

Rock-Paper-Scissors

- In this game two people (person and computer) compete
- At the same count each person displays a hand in one of three shapes:
  – Two fingers protruding – scissors
  – Flat hand – paper
  – Fist – rock
- If both shapes are the same, the game is a tie, otherwise the following rules apply

Rock-Paper-Scissors Compact Solution

One Player	Other Player	Winner
paper	rock	paper
paper	scissors	scissors
rock	scissors	rock

Let's model this game in English

An English Solution for RPS

- Both players decide how many fingers to display
- Players display their choices
- If the choices are equal, the game is a tie!
- If Player1 has paper, then Player1 wins if Player2 has rock, otherwise Player1 loses
- If Player1 has rock, the Player1 wins if Player2 has scissors, otherwise Player1 loses
- If Player2 has paper, Player1 wins, otherwise Player1 loses

Rock-Paper-Scissors Started…

```
p1 = input('Please enter p, r or s for first player. ')
p2 = input('Please enter p, r or s for second player. ')
if p1 == p2:
    print ('Game is a tie!','P1 =',p1,'P2 =',p2)
else:
    if p1 == 'p':
        if p2 == 'r':
            print ('P1 wins','P1 =',p1,'P2 =',p2)
        else:
            print ('P2 wins!','P1 =',p1,'P2 =',p2)
```

Completed English -> Python

```
p1 = input('Please enter p, r or s for first player. ')
p2 = input('Please enter p, r or s for second player. ')
if p1 == p2:
    print ('Game is a tie!','P1 =',p1,'P2 =',p2)
else:
    if p1 == 'p':
        if p2 == 'r':
            print ('P1 wins','P1 =',p1,'P2 =',p2)
        else:
            print ('P2 wins!','P1 =',p1,'P2 =',p2)
    elif p1 == 'r':
        if p2 == 's':
            print ('P1 wins','P1 =',p1,'P2 =',p2)
        else:
            print ('P2 wins!','P1 =',p1,'P2 =',p2)
    else:
        if p2 == 'p':
            print ('P1 wins','P1 =',p1,'P2 =',p2)
        else:
            print ('P2 wins!','P1 =',p1,'P2 =',p2)
```

Question

- To completely test this program, how many cases do you need to consider?

Answer

- You need to consider 9 cases: (p,p), (p,r), (p,s), (r,p),(r,r),(r,s), (s,p), (s,r), (s,s)

```
>>>
Game is a tie! P1 = p P2 = p
P1 wins P1 = p P2 = r
P2 wins! P1 = p P2 = s
P2 wins! P1 = r P2 = p
Game is a tie! P1 = r P2 = r
P1 wins P1 = r P2 = s
P1 wins P1 = s P2 = p
P2 wins! P1 = s P2 = r
Game is a tie! P1 = s P2 = s
>>>
```

Let's see how we can test this efficiently

First Look at Functions

```
def RockPaperScissors(p1,p2):
    if p1 == p2:
        print ('Game is a tie!','P1 =',p1,'P2 =',p2)
    else:
        if p1 == 'p':
            if p2 == 'r':
                print ('P1 wins','P1 =',p1,'P2 =',p2)
            else:
                print ('P2 wins!','P1 =',p1,'P2 =',p2)
        elif p1 == 'r':
            if p2 == 's':
                print ('P1 wins','P1 =',p1,'P2 =',p2)
            else:
                print ('P2 wins!','P1 =',p1,'P2 =',p2)
        else:
            if p2 == 'p':
                print ('P1 wins','P1 =',p1,'P2 =',p2)
            else:
                print ('P2 wins!','P1 =',p1,'P2 =',p2)
for p1 in 'prs':
    for p2 in 'prs':
        RockPaperScissors(p1,p2)
```

Python Modules

- Python comes with many "modules" (programs) that expand its capabilities
- To include a module named foo you just put the statement:
  `import foo`
- at the start of your program and use the functions that you want to use
- We will illustrate by showing how to make it so you can play against the computer in Rock-Paper-Scissors

Python Modules

- Documentation is found on python.org for all of the standard modules
- People can write their own, and then documentation can be embedded in the module

```
import random
def RockPaperScissors(computer,human):
    if computer == human:
        print ('Game is a tie!','computer =',computer,'human =',human)
    else:
        if computer == 'p':
            if human == 'r':
                print ('computer wins! ','computer =',computer,'human =',human)
            else:
                print ('human wins! ','computer =',computer,'human =',human)
        elif computer == 'r':
            if human == 's':
                print ('computer wins! ','computer =',computer,'human =',human)
            else:
                print ('human wins! ','computer =',computer,'human =',human)
        else:
            if human == 'p':
                print ('computer wins! ','computer =',computer,'human =',human)
            else:
                print ('human wins! ','computer =',computer,'human =',human)

computerChoice = random.choice('prs')
humanChoice = input('Please enter r, p or s. ')
RockPaperScissors(computerChoice, humanChoice)
```

Let's Make a Deal

- This is a very stripped down game
- Basically, there are 3 curtains
- The computer picks one of the 3, and puts a prize behind it
- Then you pick one of the three curtains
- If you guess the curtain with the prize, you win, otherwise you lose

Let's Make a Deal in Python

```
import random

car = random.choice('123')
human = input('Please select curtain 1, 2 or 3. ')

if human == car:
    print ('Congratulations! You won the prize.')
else:
    print ('Sorry! You did not win the prize. It was
behind curtain '+car+'.')
```

Let's Make a Deal With Swapping

- We will modify the program so that after you make a selection, the computer finds an empty curtain among the remaining curtains and permits you to switch to the other curtain
- For homework you will compare what happens if you take the switch every time to the strategy of never taking the swap
- What do you think the best strategy is?

Code for Let's Make a Deal #2

```
import random

choices = ['1', '2', '3']
car = random.choice(choices)
human = input('Please select curtain 1, 2 or 3. ')

choices.remove(car)
if human in choices:
    choices.remove(human)

empty = random.choice(choices)

print ("The empty curtain is",empty)
```

Let's Make a Deal #2

```
choices = ['1', '2', '3']
choices.remove(empty)
choices.remove(human)

yn = input('Do you want to swap for curtain ' +choices[0]+ '? (y/n) ')
if yn == 'y' :
    human = choices[0]
if human == car:
    print ('Congratulations! You won the prize.')
else:
    print ('Sorry! You did not win the prize. '),
    print ('It was behind curtain '+car+'.')
```

Hard to Understand!

- The program as completed contains only19 lines of code yet it is rather difficult to understand how it works
  - This is partly because the representation of the human concept "doors" is a Python list. Does **`choices[0]`** look like a door?
  - But more importantly, code like this is not explained at all:

```
choices.remove(car)
if human in choices:
    choices.remove(human)
```

Let's Make a Deal (with comments)

```
import random

#doors are labeled 1,2, and 3
choices = ['1', '2', '3']
car = random.choice(choices)
human = input('Please select curtain 1, 2 or 3. ')

# computer has determined which door has the car, and
# the human has made a choice. We need to show the human
# an empty door, so first remove the door with the car
choices.remove(car)

# this leaves two doors. If the human did NOT pick the
# car, remove this also
if human in choices:
    choices.remove(human)
```

Let's Make a Deal (w/comments p2)

```
# at this time we are left with only 1 choice (if the human did
# NOT pick the car) or 2 choices (if they did pick the car). In
# either case random.choice() will give us an empty door.
showDoor = random.choice(choices)

# show it to the human
print ("The empty curtain is",showDoor)

# Since there were 3 doors to start with, the human has picked
# one, and we've shown an empty door, we can only offer ONE door
# to the human to swap, which is the single door which we did
# NOT show and was NOT chosen by the human. So remove both.

choices = ['1', '2', '3']
choices.remove(showDoor)
choices.remove(human)
```

Let's Make a Deal (w/comments p3)

```
# because the choices list is numbered 0,1,2 and we've removed
# two of them, the one remaining is choices [0]

yn = input('Do you want to swap for curtain ' + choices[0]+ '?
(y/n) ')
if yn == 'y' :
    human = choices[0]
if human == car:
    print ('Congratulations! You won the prize.')
else:
    print ('Sorry! You did not win the prize. '),
    print ('It was behind curtain '+car+'.')
```

The Monty Hall Problem

- The program we just looked at demonstrates the Monty Hall problem
- If you <u>always</u> swap, you will win 2/3 of the time
- This is completely counter-intuitive, but demonstrating that this is true would take too much time … alas.