Getting Started With Python Programming: Part 2

- Getting information from the user (input)
- How information is stored, converting between different types
- Formatting text output

Input

- The computer program getting *string information* from the user.
- Strings cannot be used for calculations (information for getting numeric input will provided shortly).

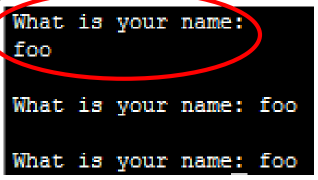
- **Format:**

```
<variable name> = input()
         OR
<variable name> = input("<Prompting message>")
```

Avoid alignment issues such as this

- **Name of the full example:** `8input.py`

```
print("What is your name: ")
name = input()
        OR
name = input("What is your name: ")
        OR
print("What is your name: ", end="")
name = input()
```



```
What is your name:
foo

What is your name: foo

What is your name: foo
```

James Tam

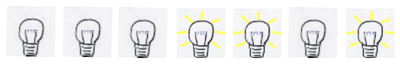
Variables: Storing Information (If There Is Time)

- On the computer all information is stored in binary (2 states)
  - Example: RAM/memory stores information in a series of on-off combinations
  - A single off/off combination is referred to as a ‘bit’

Bit

Byte

•8 bits



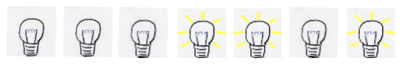
James Tam

Variables: Storing Information (If There Is Time)

- Information must be converted into binary to be stored on a computer.

User enters $\longrightarrow$ Can be stored as

13

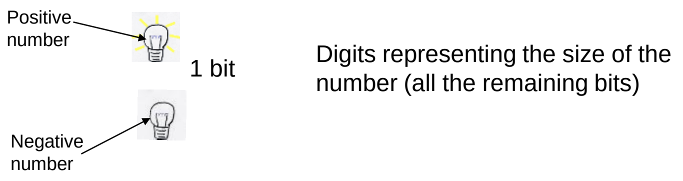


slide 4

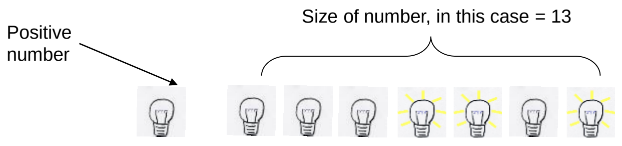
James Tam

Storing Integer Information (If There Is Time)

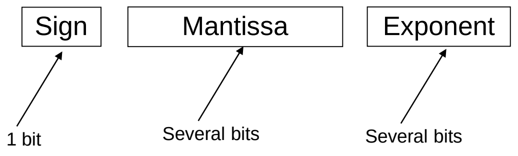
- 1 bit is used to represent the sign, the rest is used to store the size of the number
  - Sign bit: 1/on = negative, 0/off = positive
- **Format:**


Positive number
1 bit
Negative number
Digits representing the size of the number (all the remaining bits)

- **Previous example**



Storing Real Numbers In The Form Of Floating Point (If There Is Time)



- Mantissa: digits of the number being stored
- Exponent: the direction (negative = left, positive=right) and the number of places the decimal point must move ('float') when storing the real number as a floating point value.

- Examples with 5 digits used to represent the mantissa:
  - e.g. One: 123.45 is represented as 12345 * 10^{-2}
  - e.g. Two: 0.12 is represented as 12000 * 10^{-5}
  - e.g. Three: 123456 is represented as 12345 * 10^{1}
- Remember: Using floating point numbers may result in a loss of accuracy (the float is an approximation of the real value to be stored).

Storing Character Information (If There Is Time)

- Typically characters are encoded using ASCII
- Each character is mapped to a numeric value
  - E.g., ‘A’ = 65, ‘B’ = 66, ‘a’ = 97, ‘2’ = 50
- These numeric values are stored in the computer using binary

Character	ASCII numeric code	Binary code
‘A’	65	01000001
‘B’	66	01000010
‘a’	97	01100001
‘2’	50	00110010

James Tam

Storing Information: Bottom Line

- Why it important to know that different types of information is stored differently?
  - One motivation: sometimes students don’t why it’s significant that “123” is not the same as the number 123.
  - Certain operations only apply to certain types of information and can produce errors or unexpected results when applied to other types of information.
- **Example**

```
num = input("Enter a number")
numHalved = num / 2
```

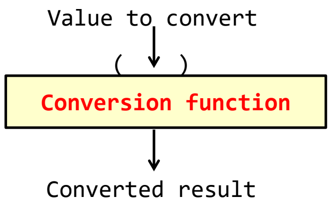
James Tam

Converting Between Different Types Of Information

- Example motivation: you may want numerical information to be stored as a string (for built in string functions e.g., check if a string consists only of numbers) but also you want to perform calculations).
- Some of the conversion mechanisms (functions) available in Python:

Format:

```
int(<value to convert>)
float(<value to convert>)
str(<value to convert>)
```



Examples:

Name of the full example: `9convert.py`

```
var1 = 10.9
var2 = int(var1)
print(var1,var2)
```

```
10.9 10
```

Converting Between Different Types Of Information (2)

Examples:

Name of the full example: `10convert.py`

```
var1 = "100"
var2 = "-10.5"
print(var1 + var2)
print(int(var1) + float(var2))
```

```
100-10.5
89.5
```

Converting Types: Extra Practice For Students

- Determine the output of the following program:

```
print(12+33)
print("12"+"33")
x = 12
y = 21
print(x+y)
print(str(x)+str(y))
```

James Tam

Converting Between Different Types Of Information: Getting Numeric Input

- The 'input()' function only returns string information so the value returned must be converted to the appropriate type as needed.
  - **Name of the full example:** 11convert.py

```
# No conversion performed: problem!
HUMAN_CAT_AGE_RATIO = 7
age = input("What is your age in years: ")
catAge = age * HUMAN_CAT_AGE_RATIO
print ("Age in cat years: ", catAge)
```

- **'Age' refers to a string not a number.**
- **The '*' is not mathematical multiplication**

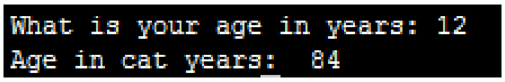
```
What is your age in years: 12
Age in cat years:  12121212121212
```

James Tam

Converting Between Different Types Of Information: Getting Numeric Input (2)

```
# Input converted: Problem solved!
HUMAN_CAT_AGE_RATIO = 7
age = int(input("What is your age in years: "))
catAge = age * HUMAN_CAT_AGE_RATIO
print("Age in cat years: ", catAge)
```

- 'Age' converted to an integer.
- The '*' now multiplies a numeric value.



```
What is your age in years: 12
Age in cat years:  84
```

James Tam

Section Summary: Input, Representations

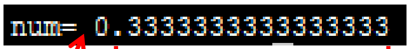
- How to get user input in Python
- How do the different types of variables store/represent information (optional/extra for now)
- How/why to convert between different types

James Tam

By Default Output Is Unformatted

- Example:

```
num = 1/3
print("num=",num)
```



```
num= 0.3333333333333333
```

Sometimes you get extra spaces (or blank lines)

The number of places of precision is determined by the language not the programmer

- There may be other issues e.g., you want to display output in columns of fixed width, or right/left aligned output
- There may be times that specific precision is needed in the displaying of floating point values

James Tam

Formatting Output

- Output can be formatted in Python through the use of **format specifiers** and **escape codes**

James Tam

Format Specifiers (If There Is Time)

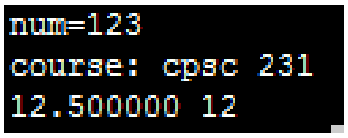
- **Format**:

```
print ("%<type of info to display/code>" %<source of the info
  to display>)
```

Doesn't literally display this: Placeholder (for information to be displayed)

- **Example (starting with simple cases)**:
  - **Name of the full example:** `12formatting.py`

```
num = 123
st = "cpsc 231"
print("num=%d"       %num)
print("course: %s"   %st)
num = 12.5
print("%f %d" %(num,num))
```



```
num=123
course: cpsc 231
12.500000 12
```

James Tam

Types Of Information That Can Be Formatted Via Format Specifiers (Placeholder) (If There Is Time)

Specifier	Type of Information to display
%s	String
%d	Integer (d = decimal / base 10)
%f	Floating point

James Tam

Formatting Effects Using Format Specifiers (If There Is Time)

- **Format**:

  `%<width>`[1]`.<precision>`[2]`<type of information>`

- **Examples (format specifiers to format output)**:
  - **Name of the full example**: `13formatting.p`

```
num = 12.55
print ("%4.1f" %num)
print ("%5.1f" %num)
print ("%3.1f" %num)
print ("%3s%-3s" %("ab", "ab"))
print ("%-3s%3s" %("ab", "ab"))
```

```
12.6
 12.6
12.6
 abab
ab  ab
```

1 A positive integer will add leading spaces (right align), negatives will add trailing spaces (left align). Excluding a value will set the field width to a value large enough to display the output

2 For floating point data only.

James Tam

One Application Of Format Specifiers (If There Is Time)

- It can be used to align columns of text.
- Example (movie credits, tabular or financial information)

James Tam

Section Summary: Formatting Output (If There Is Time)

- How to use format specifiers (field width, precision) to format output

James Tam

Escape Codes/Characters

- The back-slash character enclosed within quotes won't be displayed but instead indicates that a formatting (escape) code will follow the slash:

Escape sequence	Description
\a	Alarm: Causes the program to beep.
\n	Newline: Moves the cursor to beginning of the next line.
\t	Tab: Moves the cursor forward one tab stop.
\'	Single quote: Prints a single quote.
\"	Double quote: Prints a double quote.
\	Backslash: Prints one backslash.

James Tam

Percent Sign[1] (If There Is Time)

- If no format specifiers are used then simply enclose the '%' within the quotes of a `print()` statement

  `print("12%")` → 12%

- If format specifiers are used within a call to `print()` then use one percent sign to act as an escape code for another percent sign to follow

  `print("%f%%" %(100))` → 100.000000%

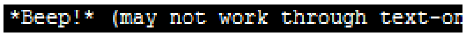
1 Since the question inevitably comes up each term I'm answering it here

James Tam

Escape Codes (2)

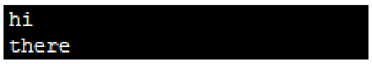
- **Program name:** `14formatting.py`

```
print ("\a*Beep!*")
```



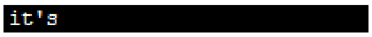
```
*Beep!* (may not work through text-on
```

```
print ("hi\nthere")
```



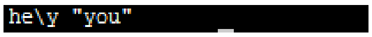
```
hi
there
```

```
print ('it\'s')
```



```
it's
```

```
print ("he\\y \"you\"")
```



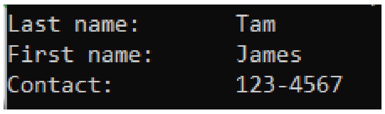
```
he\y "you"
```

James Tam

Escape Codes: Application

- It can be used to nicely format text output (alignment output, provide separators within and between lines)
- **Program example**: `15formatting.py`

```
firstName = "James"
lastName = "Tam"
mobile = "123-4567"
print("Last name:\t", lastName)
print("First name:\t", firstName)
print("Contact:\t", mobile)
```



```
Last name:        Tam
First name:       James
Contact:          123-4567
```

- Escape codes for aligning text is even more valuable if the width of a field (data to be displayed) is variable e.g., comes from user input or a text file.

James Tam

Section Summary: Escape Codes

- How to use escape codes to format output

James Tam

Extra Practice

- Traces:
  - Modify the examples (output using format specifiers and escape codes) so that they are still valid Python statements.
    - Alternatively you can try finding some simple ones online or from a textbook.
  - Hand trace the code (execute on paper) without running the program.
  - Then run the program and compare the actual vs. expected result.
- Program writing:
  - Write a program the will right-align text into 3 columns of data.
  - Write a program the will left-align text into 3 columns of data.

James Tam

After This Section You Should Now Know

- How to format output through:
  - The use of format specifiers
  - Escape codes

James Tam