Lecture 6 – Getting user input, converting between types, generating random numbers

**Learning outcomes**

- At the end of this lecture, students should be able to:
  - get user input from the keyboard
  - generate a random number
  - convert between types

**Recap**

- From lecture 5
  - use dot notation when using string methods with string instances
  - use string methods: upper(), lower(), strip(), find(), rfind()
  - use the inbuilt functions: min(), max(), round()

```python
phrase = "When in doubt, mumble."
pos1 = phrase.find("in")
pos2 = phrase.rfind("mumb")
pos3 = phrase.rfind("ni")
total = pos1 + pos2 + pos3
print("1. Total: ", total)
phrase_lower = phrase.lower()
print("2. ", phrase_lower[:3])
smallest = min(32.7, 56.4, 3, -1.1, 56.99, -1.2)
largest = max(32.7, 56.4, 3, -1.1, 56.99, -1.2)
um1 = 32.657123
print("3. ", round(num1))
print("4. ", round(num1, 2))
```

1. Total: 19
2. whe
3. 33
4. 32.66

**Getting input from the user**

- We have already seen how the print() function is used to print to the standard output. We would now like our programs to be able to get input from the user from the keyboard (the standard input).

- The **input()** function is used to get information from the user.

This function displays the prompt, waits for the user to type their information and, as soon as the user presses the 'Enter' key, the input() function returns the information typed by the user (to the variable on the left of the assignment operator).

```python
user_name = input("Enter name: ")
colour = input("Enter colour: ")
user_word = input("Enter word: ")
print(user_name, "entered", colour, "and the word", user_word)
```

Enter name: Adriana
Enter colour: magenta
Enter word: parola
Adriana entered magenta and the word parola

The user input is shown in a pink colour
Getting input from the user

- The **input() function** can be used with no argument (nothing inside the round brackets) in which case no prompt is displayed.

- The input() function always returns a **string**. The end of line character is not returned as part of the returned string.

```python
user_number = input("Enter number: ")
user_input = input()
print("You entered", user_number, "and then", user_input)
```

```python
Enter number: 98
??? #user enters stuff here
You entered 98 and then ???
```

Ooops!

- The following code causes an error:

```python
age = input("Enter age: ")
years_to_go = 100 - age
print("Big birthday coming up in ", years_to_go, "years!")
```

```python
Enter age: 54
File "LectureCode.py", line 2, in <module>
  years_to_go = 100 - age
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

- What does this error message mean?

Converting between types

- The subtraction operator (-) has no meaning if one of the operands is a string. We want to find a way of converting a string containing just digits into a number.

- The **int() function** converts a string containing characters which are digits into the corresponding integer value.

```python
age = input("Enter age: ")
age = int(age)
years_to_go = 100 - age
print("Big birthday coming up in ", years_to_go, "years!")
```

```python
Enter age: 54
Big birthday coming up in 46 years!
```

- Note that the code on lines 1 and 2 can be combined into one line:

```python
age = int(input("Enter age: "))
```

Exercise

- Complete the output if the user enters the number 54 when prompted:

```python
user_number = input("Enter number: ")
print(user_number * 2, user_number * 3, user_number * 4)
```

```python
Enter number: 54
```

```python
Ooops!
```

- The following code causes an error:

```python
age = input("Enter age: ")
years_to_go = 100 - age
print("Big birthday coming up in ", years_to_go, "years!")
```

```python
Enter age: 54
File "LectureCode.py", line 2, in <module>
  years_to_go = 100 - age
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

- What does this error message mean?
Converting between types

- Other functions which can be used to convert between types:
  - `float()`
  - `str()`

```python
cost = input("Enter cost $")
cost = float(cost)
final_price = cost * 0.92
print("Final price $", final_price, sep="")
Enter cost $509.59
Final price $468.8228
```

- String concatenation requires that the two operands be strings:

```python
price = 19.99
message = "The price of the item is: $" + str(price)
print(message)
The price of the item is $19.99
```

Random numbers

- Quite often, in our programs, we need to generate random numbers, e.g., for games and simulations.

  - The random module contains a function, `randrange()`, which can be used to generate a random number. In order to use this function we need to import the random module into our program (just as we did when we wanted to use the functions defined in the math module - `math.sin()`, `math.cos()`, ...).

  - Whenever we need to get a random number in a program, the first line of the program will be the following import statement:

```python
import random
```

```python
dice1 = random.randrange(1, 7)
dice2 = random.randrange(1, 7)
print("You threw", dice1, "and a", dice2)
You threw 4 and a 1
```
Exercise

Complete the following program so that it prompts the user for their first name. The program removes a random letter from the first name and prints the resulting name.

```python
import random
first_name = input("Enter name: ")
print(first_name)
```

Enter name: Adriana
Adriaa
Enter name: Eric
ric
Enter name: Jerry
Jery

Exercise

Give the smallest and the largest possible random number which can be generated by the following four statements:

```python
import random
print("1.", random.randrange(4, 17, 3))
print("2.", random.randrange(-1, 7, 2))
print("3.", random.randrange(100, 700, 2))
print("4.", random.randrange(50, 100, 10))
```

Random numbers

The `randrange()` function has an optional step argument:

```
random.randrange(start, end, step)
```

When the step argument is omitted the default step is 1. The step argument dictates which random numbers are generated within the range (the step is the amount between each random number generated), e.g.,

```
random.randrange(1, 7, 2) #generates either 1, 3 or 5
random.randrange(1, 7, 3) #generates either 1 or 4
random.randrange(78, 100, 7) #generates either 78, 85, 92, or 99
random.randrange(-30, 100, 30) #generates either -30, 0, 30, 60, or 90
```

Line continuation

The preferred way of line continuation (breaking long lines of code over multiple lines) is to use parentheses, brackets and braces.

```python
user_dice = input("Enter dice throw: ")
comp_dice = random.randrange(1, 7)
message = "Your dice: " + str(user_dice) + ", computer dice: " + str(comp_dice)
print(message)
```

There will be other examples of this in later lectures.

Note that the backslash is the last character on the line of code.
Exercise

Complete the following program which prompts the user once for the total of 4 dice throws, the program calculates the sum of four random dice throws, and outputs the four dice values and the difference between the user guess and the dice throwing simulation sum (see example outputs):

```python
prompt = 'Enter sum (4 dice): '
dice_sum = 0
the_dice = "(*)

print("The dice: ", )
print("Your total: ", )
print("You are out by: ", )

Enter sum (4 dice): 12
The dice: (3 4 3 2)
Your total: 12 Dice total: 12
You are out by: 0

Enter sum (4 dice): 15
The dice: (3 1 4 6)
Your total: 15 Dice total: 14
You are out by: 1

Enter sum (4 dice): 12
The dice: (6 3 6 4)
Your total: 12 Dice total: 19
You are out by: 7
```

Summary

In a Python program:

- the `input()` function is used to get user input from the keyboard
- a random number can be generated using `random.randrange(...)`
- we can convert between types using `str()`, `int()`, `float()`

Examples of Python features used in this lecture

```python
dice1 = random.randrange(1, 7)
age = random.randrange(66, 99)
even_number = random.randrange(50, 99, 2)
tens = random.randrange(50, 101, 10)

user_input = input("Enter age: ")
age = int(user_input)

cost = input("Enter cost $")
cost = float(cost)

price = 32.45
message = "Final price $" + str(price)
```