CS177 Python Programming

Recitation 2 - Computing with Numbers
Outline

• Data types.
• Variables
• Math library.
• Range Function
What is data (in the context of programming)?

- Values that are stored and manipulated by computer programs.
- In Python, data can be: numbers, strings (character, word, sentence, etc.), or objects (will be discussed in the later chapter)
Numeric Data Type

- Numeric is the most common data type
- Numbers can be classified as natural numbers, integers, rational numbers, real numbers, complex numbers, computable numbers ...
Numbers in Python

• In Python, whole numbers are referred to *integers*; fractions are referred to *floating point*
  – A numeric literal **WITHOUT** a decimal point produces an int value
  – A literal that **WITH** a decimal point is represented by a float (even if the fractional part is 0)
String Data Type in Python

• A sequence of characters e.g., “Programming is cool!”

• Strings are encloses with
  – single quotes ‘this is string’
  – double quotes “This is a string”
  – triple double quote “““This is a string”””

• Two useful functions are used with Strings are
  input() and eval()
String Data Type in Python

• `input()` is used to obtain data from user
  - Example

```python
>>>applicant = input("Enter the applicant's name: ")
>>>interviewer = input("Enter the interviewer's name: ")
>>>time = input("Enter the appointment time: ")
>>>print(interviewer, "will interview", applicant, "at", time)
```
String Data Type in Python

- `eval(expression)`
- The `expression` argument is parsed and evaluated as a Python expression

- Example

```python
>>> x = input("Enter the value of x: ")
Enter the value of x: 1
print(eval(x))
```

Result: 1
String Data Type in Python

- We can perform + operation within `eval()`
  
  - Example
  
  ```python
  x = "10"
  print(eval(x+"1"))
  ```
  
  Result: 101

- Notice that the `expression` argument should be a string type
Arithmetic Operators

• Arithmetic operators inherit their definitions on numerical data types (int/floating point).
• Operations on floats produce floats.
• Operations on int produce int (except for /).
• What if one operand is int and the other is floating point?
  – 2+3.0
Variables

• Are names for objects that have values in main memory.
• We define variables by assigning values to names:
  Example:
  >>>a = 9
  >>>b = 13.54
  >>>c = “Computer”

• Variables can change their values during the program.

• It’s important to choose **good** names that describes the variable. Example: firstName, courseNo, etc.
Assignment Statement

Example:

\[ X = 10 \]

- Variable is also called **Left Hand Side**.
- Value is also called **Right Hand Side**.
Assignment Statement

- The value can be: a number, a string of characters, but also a variable, or an expression.
- Example:
  
  \[
  \begin{align*}
  X &= 10 \\
  Y &= X \\
  Y &= X + 5 \\
  Z &= Y
  \end{align*}
  \]
Example 1

What is the output of the following statements:

```python
>>> X = 10
>>> Y = 20
>>> result = X + Y
>>> print(result)  # 30
>>> Y = 100
>>> print(x + 10)  # 10
>>> print(x)       # 10
>>> print(y)       # 100
>>> print(result)  # 30
```
Exercise 1

```python
>>> name = "John"
>>> price = 9.99
>>> total = price / 3.0
>>> name = "Jane"
>>> print(name)
>>> print(price)
>>> print(total)
```
Exercise 2

```python
code
>>> var1 = "Purdue"
>>> var2 = "University"
>>> print(var1)
>>> print(var2)
>>> var2 = var1
>>> print(var1)
>>> print(var2)
```

Output:
```
Purdue
University
Purdue
Purdue
```
What about ...

```python
>>> X + 2 = 15
>>> Y = “Hello” + 10
```

Tips:

- Left Hand Side should always be a variable
- You can’t perform operations (such as sum) on two values of different types (numeric and string)
Math Library

• Python has math library that can perform powerful computations
  – *time*: provides various time-related functions
  – *math*: provides access to the mathematical functions defined by the C standard
  – *random*: generates pseudo-random numbers with various common distributions.
Math Library

• before using a library, we need to import the library into our program:

• Example
  – import math

• Then we can do...
  math.pow(x, y) - Return x raised to the power y.
  math.sqrt(x) - Return the square root of x.
  math.factorial(x) - Return x factorial.
  math.ceil(x) - Return the ceiling of x.
  math.floor(x) - Return the floor of x.
Math Library Examples

```python
>>> import math
>>> a = math.factorial(6)
>>> print(a)
720

>>> b = math.sqrt(123)
>>> print(b)
11.0905365064
```
Math Library Examples

>>> c = math.floor(5.9)
>>> print(c)
5

>>> x = math.factorial(4) * math.pow(2, 3)
>>> print(x)
192.0
Math Library Examples

>>> y = 5.5
>>> z = math.floor(y) * math.ceil(y)
>>> print(z)

>>> y = -5.5
>>> z = math.floor(y) * math.ceil(y)
>>> print(z)

The results for both code snippets are 30, but does math.floor(5.5) equal to abs(math.floor(-5.5))?
Function `range`

- Range is a function that returns a sequence
- If `range` has only one input parameter: (i.e `range(input)`)
  - It generates the sequence of all the non-negative integers that are less than the `input` parameter value
  - the generated sequence starts with 0
  - increment is 1
  - the last element of the sequence is the value of `input` parameter – 1

```python
>>> list(range(3))
[0, 1, 2]

>>> list(range(1))
[0]

>>> list(range(-1))
[]

>>> list(range(9))
[0, 1, 2, 3, 4, 5, 6, 7, 8]

>>> list(range(0))
[]

>>> list(range(-5))
[]
```
Function `range` 

- If two inputs (i.e. `range(first_input, second_input)`):
  - It generates the sequence of all the integers that are greater than or equal to the `first_input` value and less than the `second_input` value.
  - The first element of the sequence is the value of `first_input`.
  - Increment is 1.
  - The last element of the sequence is the value of `second_input − 1`.

```python
>>> list(range(0, 3))
[0, 1, 2]
>>> list(range(4, 7))
[4, 5, 6]
>>> list(range(-2, 2))
[-2, -1, 0, 1]
>>> list(range(0, 10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(7, 4))
[]
>>> list(range(-2, -5))
[]
```
Function `range`

- If three inputs (i.e. `range(first_input, second_input, third_input)`):
  - the sequence starts with the *first_input* value
  - increment is *third-input*
  - If increment is positive the sequence ends with the largest value less than *second_input*
  - If increment is negative the sequence ends with the smallest value greater than *second_input*

```python
>>> list(range(0, 3, 1))
[0, 1, 2]

>>> list(range(0, 6, 3))
[0, 3]

>>> list(range(1, 7, 2))
[1, 3, 5]

>>> list(range(-7, -1, 2))
[-7, -5, -3]

>>> list(range(7, 1, -2))
[7, 5, 3]

>>> list(range(-5, 5, 3))
[-5, -2, 1, 4]
```
Exercises

type(9.5)
type(2)
type(2.0)
type(0.0)
type(0)
Exercises

type(2)
type(0.0)
type(2+3)
type(2.0+3)
type(2*3)
type(2/3)
type(2//3)
type(2%3)
Questions from past exams

• What is the output of the following Python program?
  
x = list(range(7, 1, -2))
  print(x[-2])

A. 1
B. 2
C. 3
D. 5
E. 7
Questions from past exams

• What is the result of evaluating the following expression $2^{**4}+9/3*2-2$?
  
  A. 2.166666667
  
  B. 14.666666668
  
  C. 20.0
  
  D. 36.0
  
  E. None of the above
Questions from past exams

• What is the output of the following Python program?
  ```python
def testFun(a, b, c):
    print(a+b+c)

testFun(11, 12, ‘13’)
  ```

A. 36  B. 2313  C. ‘2313’  D. 23+’13’  E. TypeError
Questions from past exams

• What is the output of the following Python program?

def func(x, y):
    y = y*2
    x = x+y
    return x

print(func(‘1’, ‘2’))

A. 5
B. 23
C. 14
D. 122
E. 211