CS177 Python Programming

Recitation 2 - Computing with Numbers
Outline

• Data types.
• Variables
• Math library.
• Range Function
What is data in a programming context?

• Values that are stored and manipulated by computer programs.

• In Python, data can be:
  – Numbers,
  – Strings (character, word, sentence, etc.)
  – Objects (will be discussed in the later chapter)
Why does data types matter?

• It defines the way data is stored in memory

• It defines the operations you can apply to the data

• We will see in a later recitation mutable and immutable data, expected to master in this course.
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 as integers (key word: `int`); fractions are referred to as floating point (key word: `float`)

  – 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(). How?
String Data Type in Python

• `input()` is used to obtain data from user
  – Example:
    ```
    >>> 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)
    ```

• Whatever the user enter is of type `string` by default.

• How can we change the type? By either using `eval()` or making a `cast` to the desired type.
String Data Type in Python

eval() vs casting:

When casting we can use any desired type: `int()`, `float()`, `str()`
String Data Type in Python

`eval()` vs `casting`:

- `eval(expression)`
- `int(expression)`

We can combine the two functions in one line!

We highly recommend using casting instead of `eval()`
String Data Type in Python

• We can concatenate two strings using the operator +
• The resulting string is a completely new string, stored in a different position in memory
• That means that is a completely new variable in memory. **Important for exams!!!**
String Data Type in Python

• Let’s see some examples

```python
def main():
    str1 = "Hello"
    str2 = 'World!'
    str3 = str1 + " " + str2
    print(str3)

main()
```

The result would be?

```python
def main():
    value1 = "10"
    value2 = "1"
    result = value1 + value2
    print(result)

main()
```

```plaintext
>>> ========
>>> 101
```
Arithmetic Operators

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

```python
>>> 3 + 3
6
>>> 3 + 3.0
6.0
>>> 3+4*3
15
```

Precedence is important. **MASTER** this for the exams!
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

Variable ← Value

Example:
X = 10

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
</table>

- The value can be: a number, a string of characters, **but also a variable, or an expression**.
- Example:
  - \( X = 10 \)
  - \( Y = X \)
  - \( Y = X + 5 \)
  - \( Z = Y \)
Example 1:

What is the output of the following statements:

```python
>>> X = 10
>>> Y = 20
>>> result = X + Y
>>> print(result)
>>> Y = 100
>>> print(X + 10)
>>> print(X)
>>> print(X)
```

Output:

<table>
<thead>
<tr>
<th>MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>
Example 2:

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

Output:

```
Jane
9.99
3.33
```
Example 3:

```python
>>> var1 = "Purdue"
>>> var2 = "University"

>>> print(var1)
"Purdue"

>>> print(var2)
"University"

>>> var2 = var1

>>> print(var1)
"Purdue"

>>> print(var2)
"Purdue"
```

Output:
```
Purdue
University
Purdue
```

With respect to the MEMORY `int`, `float` and `strings` behave in this way because they are **mutable**. Further explanation in a later recitation.
What about ...

>>> 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

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

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

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

C. 20.0
Questions from past exams

• What is the output of the following Python program?

def main():
    a = 11
    b = 12
    c = "13"
    print(a+b+c)

main()

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