DATA 301

Python II

Dr. Firas Moosvi
University of British Columbia Okanagan
firas.moosvi@ubc.ca

Python Math Expressions

Math ***expressions*** in Python:

Operation	Syntax	Example
Add	+	5 + 3
Subtract	-	10 – 2
Multiply	*	5 * 3
Divide	/	9 / 4
Modulus	%	9 % 4 (answer is 1)
Exponent	**	5 ** 2 (answer is 25)

Expressions - Operator Precedence

Each operator has its own priority similar to their priority in regular math expressions:

- 1) Any expression in parentheses is evaluated first starting with the inner most nesting of parentheses.
- 2) Exponents
- 3) Multiplication and division (*, /, %)
- 4) Addition and subtraction (+,-)

Python Expressions Question

Question: What is the value of this expression:

```
8 ** 2 + 12 / 4 * (3 - 1) % 5
```

A) 69 **B)** 65 **C)** 36 **D)** 16 **E)** 0

Try it: Python Variables and Expressions

Question 1: Write a program that prints the result of `35 + 5 * 10`.

Question 2: Write a program that uses at least 3 operators to end up with the value `99`.

Question 3: Write a program that has a variable called `name` with the value of your name and a variable called `age` storing your age. Print out your name and age using these variables.

Strings

Strings are sequences of characters that are surrounded by either single or double quotes.

- Use \ to escape ' E.g. There\'s
- Can use triple double quotes """ for a string that spans multiple lines.

Example:

```
name = "Joe Jones"

storeName = 'Joe\'s Store'
print("""String that is really long
with multiple lines
   and spaces is perfectly fine""")
```

Python String Indexing

Individual characters of a string can be accessed using square brackets (`[]`) with the first character at index 0.

Example:

```
str = "Hello"
print(str[1])          # e
print("ABCD"[0])       # A
print(str[-1])         # o
# Negative values start at end and go backward
```

Rules for Strings in Python

Must be surrounded by single or double quotes.

Can contain most characters except enter, backspace, tab, and backslash.

- These special characters must be escaped by using an initial "\".
- e.g. `\n` – new line, `\'` – single quote, `\\` - backslash, `\"` – double quote
- A string in raw mode (`r` before quote) will ignore backslash escape. May be useful if data contains escapes. Example: `st = r"slash\there\"`

Double quoted strings can contain single quoted strings and vice versa.

Any number of characters is allowed.

The minimum number of characters is zero `""`, which is called the *empty string*.

String *literals* (values) have the quotation marks removed when displayed.

Python Strings Question

Question: How many of the following are valid Python strings?

1) `""`
2) `''`
3) `"a"`
4) `"  "`
5) `"""`
6) `"Joe\' Smith\""`

A) 1 **B)** 2 **C)** 3 **D)** 4 **E)** 5

Python String Functions

```
st  = "Hello"
st2 = "Goodbye"
```

Operation	Syntax	Example	Output
Length	`len()`	`len(st)`	`5`
Upper case	`upper()`	`st.upper()`	`HELLO`
Lower case	`lower()`	`st.lower()`	`hello`
Convert to a string	`str()`	`str(9)`	`"9"`
Concatenation	`+`	`st1 + st2`	`HelloGoodbye`
Substring	`[]`	`st[0:3]` `st[1:]`	`Hel` `ello`
String to int	`int()`	`int("99")`	`99`

String Operators: Concatenation

The ***concatenation operator*** is used to combine two strings into a single string. The notation is a plus sign '**+**'.

Example:

```
st1 = "Hello"
st2 = "World!"
st3 = st1 + st2 # HelloWorld!
print(st1+st1)
num = 5
print(st1+str(num))    # Hello5
# Must convert number to string before
# concatenation
```

String Concatenation Question

Question: What is the output of this code?

```
st1 = "Hello"
st2 = "World!"
num = 5
print(st1 + str(num) + " " + st2)
```

A) `Error`

B) `Hello5World!`

C) `Hello5 World!`

D) `Hello 5 World!`

Substring

The ***substring*** function will return a range of characters from a string.

Syntax:

```
st[start:end]   # start is included, end is not
                # first character is index 0
```

Examples:

```
st = "Fantastic"
print(st[1])          # a
print(st[0:6])        # Fantas
print(st[4:])         # astic
print(st[:5])         # Fanta
print(st[-6:-2])      # tast
```

Substring Question

Question: What is the output of this code?

```
st = "ABCDEFG"
print(st[1] + st[2:4] + st[3:] + st[:4])
```

A) ABCDCDEFGABCD

B) ABCDEFGABC

C) ACDDEFGABCD

D) BCDDEFGABCD

E) BCDECDEFGABC

Split

The ***split*** function will divide a string based on a separator.

Examples:

```
st = "Awesome coding! Very good!"
print(st.split())
# ['Awesome', 'coding!', 'Very', 'good!']
print(st.split("!"))
# ['Awesome coding', ' Very good', '']
st = 'data,csv,100,50,,25,"use split",99'
print(st.split(","))
# ['data', 'csv', '100', '50', '', '25',
#   '"use split"', '99']
```

Try it: Python String Variables and Functions

Question 1: Write a Python program that prints out your name and age stored in variables like this:

```
Name: Joe
Age: 25
```

Question 2: Write a Python program that prints out the first name and last name of `Steve Smith` like below. You must use substring.

- Bonus challenge: Use `find()` function so that it would work with any name.

```
First Name: Steve
Last Name: Smith
```

Print Formatting

The `print` method can accept parameters for formatting.

```
print("Hi", "Amy", ", your age is", 21)
print("Hi {}, your age is {}".format("Amy",21))
```

This is one of the most obvious changes between Python 2:

```
print "Hello"
```

and Python 3:

```
print("Hello")
```

Python Date and Time

Python supports date and time data types and functions.

First, import the `datetime` module:

```
from datetime import datetime
```

Functions:

```
now = datetime.now()
print(now)
current_year = now.year
current_month = now.month
current_day = now.day
print("{}-{}-{} {}:{}:{}".format(now.year, now.month,
now.day, now.hour, now.minute, now.second))
```

Python Clock

Python `time()` function returns the current time in seconds:

```
import time
startTime = time.time()
print("Start time:", startTime)
print("How long will this take?")
endTime = time.time()
print("End time:", endTime)
print("Time elapsed:", endTime-startTime)
```

Python Input

To read from the keyboard (standard input), use the method `input`:

```
name = input("What's your name?")
print(name)
age = input("What's your age?")
print(age)
```

Prompt for value from user

print out value received

- Note in Python 2 the method is called `raw_input()`.

Try it: Python Input, Output, and Dates

Question 1: Write a program that reads a name and prints out the name, the length of the name, the first five characters of the name.

Question 2: Print out the current date in `YYYY/MM/DD` format.

Comparisons

A ***comparison operator*** compares two values. Examples:

- `5 < 10`
- `N > 5   # N is a variable. Answer depends on what is N.`

Comparison operators in Python:

- `>` - Greater than
- `>=` - Greater than or equal
- `<` - Less than
- `<=` - Less than or equal
- `==` - Equal (Note: Not "=" which is used for assignment!)
- `!=` - Not equal

The result of a comparison is a ***Boolean value*** which is either **`True`** or **`False`**.

Conditions with `and`, `or`, `not`

A ***condition*** is an expression that is either `True` or `False` and may contain one or more comparisons. Conditions may be combined using: `and`, `or`, `not`.

- order of evaluation: `not`, `and`, `or` May change order with parentheses.

Operation	Syntax	Examples	Output

AND
(True if both are True)
	`and`	`True and True`
		`False and True`
		`False and False`
			`True`
			`False`
			`False`

OR
(True if either or both are True)
	`or`	`True or True`
		`False or True`
		`False or False`
			`True`
			`True`
			`False`

NOT **(Reverses: e.g. True becomes False)**	`not`	`not True` `not False`	`False` `True`

Condition Examples

```
n = 5
v = 8
print(n > 5)                          # False
print(n == v)                         # False
print(n != v)                         # True
print(n == v and n+4>v)               # False
print(n == v or n+4>v)                # True
print(n+1 == v-2 or not v>4)          # True
```

Python Condition Question

Question: How many of the following conditions are **TRUE**?

1) `True and False`
2) `not True or not False`
3) `3 > 5 or 5 > 3 and 4 != 4`
4) `(1 < 2 or 3 > 5) and (2 == 2 and 4 != 5)`
5) `not (True or False) or True and (not False)`

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

Decisions

Decisions allow the program to perform different actions based on conditions. Python decision syntax:

```
if condition:
    statement
```

Done if condition is `True`

```
if condition:
    statement
else:
    statement
```

Done if condition is `False`

- The statement after the `if` condition is only performed if the condition is `True`.
- If there is an `else`, the statement after the else is done if condition is `False`.
- Indentation is important! Remember the colon!

Decisions `if/elif` Syntax

If there are more than two choices, use the `if/elif/else` syntax:

```
if condition:
   statement
elif condition:
   statement
elif condition:
   statement
else:
   statement
```

```
if n == 1:
   print("one")
elif n == 2:
   print("two")
elif n == 3:
   print("three")
else:
   print("Too big!")
print("Done!")
```

Decisions: Block Syntax

Statements executed after a decision in an `if` statement are indented for readability. This indentation is also how Python knows which statements are part of the block of statements to be executed.

- If you have more than one statement, make sure to indent them. Be consistent with either using tabs or spaces. Do not mix them!

```
if age > 19 and name > "N":
   print("Not a teenager")
   print("Name larger than N")
else:
   print("This is statement #1")
   print(" and here is statement #2!")
```

Question: Decisions

Question: What is the output of the following code?

```
n = 3
if n < 1:
   print("one")
elif n > 2:
   print("two")
elif n == 3:
   print("three")
```

A) nothing **B)** `one` **C)** `two` **D)** `three`

Question: Decisions (2)

Question: What is the output of the following code?

```
n = 3
if n < 1:
   print("one")
elif n > 2
   print("two")
else:
   print("three")
```

A) nothing

B) `one`

C) `two`

D) `three`

E) error

Question: Decisions (3)

Question: What is the output of the following code?

```
n = 1
if n < 1:
   print("one")
elif n > 2:
   print("two")
else:
   print("three")
print("four")
```

A) nothing

B)
```
one
four
```

C)
```
three
```

D)
```
three
four
```

E) error

Question: Decisions (4)

Question: What is the output of the following code?

```
n = 0
if n < 1:
    print("one")
    print("five")
elif n == 0:
    print("zero")
else:
    print("three")
print("four")
```

A) nothing

B)
```
one
four
```

C)
```
one
five
four
```

D)
```
one
five
zero
four
```

E) error

Try it: Decisions

Question 1: Write a Python program that asks the user for a number then prints out if it is even or odd.

Question 2: Write a Python program that asks the user for a number between 1 and 5 and prints out the word for that number (e.g. 1 is one). If the number is not in that range, print out error.