Department of Computer Science
Final Exam Summer 2020

Subject: Data Sciences
Time: 9 Am to 12:00 Am
BS (CS,SE)
Name: Changaiz khan
ID: 13206 Instructor: M.Ayub Khan

Q1. a. What are variables in python explain with help of Python coded examples?

Answer:

Python variable is a reserved memory location to store values. In other words, a variable in a python program gives data to the computer for processing.

Every value in Python has a datatype. Different data types in Python are Numbers, List, Tuple, Strings, Dictionary, etc. Variables can be declared by any name or even alphabets like a, aa, abc, etc.

Python Example

```
# Declare a variable and initialize it
f = 0
print f
# re-declaring the variable works
f = 'guru99'
print f
```

Python Example

```
# Declare a variable and initialize it
f = 0
print(f)
# re-declaring the variable works
f = 'guru99'
print(f)
```

Q1b. What are the rules to define a variable in python?

Answer:

Creating Variables

Variables are containers for storing data values.

Unlike other programming languages, Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Example

```
x = 5
y = "John"
print(x)
print(y)
```

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Example

```
#Legal variable names:
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"

#Illegal variable names:
2myvar = "John"
my-var = "John"
my var = "John"
```

Assign Value to Multiple Variables

Python allows you to assign values to multiple variables in one line:

Example

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

Output Variables

The Python `print` statement is often used to output variables.

To combine both text and a variable, Python uses the `+` character:

Example

```
x = "awesome"
print("Python is " + x)
```

Q2. a. What are data types, how many data types are used in python explain with
the help of Python coded examples ?

Answer:

Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`

Getting the Data Type

You can get the data type of any object by using the type() function:

Example

Print the data type of the variable x:

```
x = 5
print(type(x))
```

Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

x = "Hello World"	str
x = 20	int
x = 20.5	float

x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	

Setting the Specific Data Type

If you want to specify the data type, you can use the following constructor functions:

Example	Data Type
x = str("Hello World")	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list(("apple", "banana", "cherry"))	list
x = tuple(("apple", "banana", "cherry"))	tuple
x = range(6)	range
x = dict(name="John", age=36)	dict
x = set(("apple", "banana", "cherry"))	set

x = frozenset(("apple", "banana", "cherry"))	frozenset
x = bool(5)	bool
x = bytes(5)	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	

Q2b. Write a program in python in which integer value is changed in to string data type as well as explain in detail.

Answer:

To convert an integer to string in Python, use the `str()` function. This function takes any data type and converts it into a string, including integers. Use the syntax `print(str(INT))` to return the `int` as a str, or string.

Converting int to Strings

We can convert numbers to strings through using the `str()` method. We'll pass either a number or a variable into the parentheses of the method and then that numeric value will be converted into a string value.

Let's first look at converting integers. To convert the integer `12` to a string value, you can pass `12`into the `str()` method:

```
str(12)
```

Copy

When running `str(12)` in the Python interactive shell with the `python` command in a terminal window, you'll receive the following output:

```
Output
'12'
```

The quotes around the number 12 signify that the number is no longer an integer but is now a string value.

With variables we can begin to see how practical it can be to convert integers to strings. Let's say we want to keep track of a user's daily programming progress and are inputting how many lines of code they write at a time. We would like to show this feedback to the user and will be printing out string and integer values at the same time:

```
user = "Sammy"
lines = 50

print("Congratulations, " + user + "! You just wrote " + lines + " lines of code.")
```

Copy

When we run this code, we receive the following error:

```
Output
TypeError: Can't convert 'int' object to str implicitly
```

We're not able to concatenate strings and integers in Python, so we'll have to convert the variable `lines` to be a string value:

```
user = "Sammy"
lines = 50

print("Congratulations, " + user + "! You just wrote " + str(lines) + " lines of 
code.")
```

Q3. Why print() and type functions are used in python explain with the help of python coded examples for each function and explain in detail as well ?

Answer.

PRINT FUNCTION:

The `print()` function prints the specified message to the screen, or other standard output device.

The message can be a string, or any other object, the object will be converted into a string before written to the screen.

Syntax

print(*object(s)*, sep=*separator*, end=*end*, file=*file*, flush=*flush*)

Parameter	**Description**
object(s)	Any object, and as many as you like. Will be converted to string before printed
sep='*separator*'	Optional. Specify how to separate the objects, if there is more than one. Default is ' '
end='*end*'	Optional. Specify what to print at the end. Default is '\n' (line feed)
file	Optional. An object with a write method. Default is sys.stdout
flush	Optional. A Boolean, specifying if the output is flushed (True) or buffered (False). Default is False

Example

Print more than one object:

```
print("Hello", "how are you?")
```

Example

Print a tuple:

```
x = ("apple", "banana", "cherry")
print(x)
```

Example

Print two messages, and specify the separator:

```
print("Hello", "how are you?", sep="---")
```

TYPE FUNCTION:

type() method returns class type of the argument(object) passed as parameter. type() function is mostly used for debugging purposes.

Two different types of arguments can be passed to type() function, single and three argument. If single argument type(obj) is passed, it returns the type of given object. If three arguments type(name, bases, dict) is passed, it returns a new type object.

Syntax :

```
type(object)
type(name, bases, dict)
```

Parameters :

name : name of class, which later corresponds to the __name__ attribute of the class.
bases : tuple of classes from which the current class derives. Later corresponds to the __bases__ attribute.

dict : a dictionary that holds the namespaces for the class. Later corresponds to the __dict__ attribute.

Q4. How addition operator is used to update the values of variables explain with the help of Python coded example as well as explain the program?

Answer:

One of the most common forms of reassignment is an **update** where the new value of the variable depends on the old. For example,

```
x = x + 1
```

This means get the current value of x, add one, and then update x with the new value. The new value of x is the old value of x plus 1. Although this assignment statement may look a bit strange, remember that executing assignment is a two-step process. First, evaluate the right-hand side expression. Second, let the variable name on the left-hand side refer to this new resulting object. The fact that x appears on both sides does not matter. The semantics of the assignment statement makes sure that there is no confusion as to the result. The visualizer makes this very clear.

```
x = 6
x = x + 1
```

```
x = x + 1

x = 6       # initialize x

print(x)

x = x + 1    # update x

print(x)
```

If you try to update a variable that doesn't exist, you get an error because Python evaluates the expression on the right side of the assignment operator before it assigns the resulting value to the name on the left. Before you can update a variable, you have to initialize it, usually with a simple assignment. In the above example, x was initialized to 6.

Updating a variable by adding 1 is called an increment; subtracting 1 is called
a decrement. Sometimes programmers also talk about bumping a variable, which
means the same as incrementing it by 1.

Q5. What type of errors do occur in Python, write the a program with different
types of errors as well as write separate correction code in python as well as explain the errors?

Answer:

TYPES OF ERROR:

The most common reason of an error in a Python program is when a certain statement is not in accordance with the prescribed usage. Such an error is called a syntax error. The Python interpreter immediately reports it, usually along with the reason.

```
> print "hello"
SyntaxError: Missing parentheses in call to 'print'. Did you
mean print("hello")?
```

In Python 3.x, print is a built-in function and requires parentheses. The statement above violates this usage and hence syntax error is displayed.

Many times though, a program results in an error after it is run even if it doesn't have any syntax error. Such an error is a runtime error, called an exception. A number of built-in exceptions are defined in the Python library. Let's see some common error types.

IndexError is thrown when trying to access an item at an invalid index.

```
> L1=[1,2,3]
> L1[3]
Traceback (most recent call last):
File "<pyshell#18>", line 1, in <module>
```

```
L1[3]
IndexError: list index out of range
```

ModuleNotFoundError is thrown when a module could not be found.

```
> import notamodule
Traceback (most recent call last):
File "<pyshell#10>", line 1, in <module>
import notamodule
ModuleNotFoundError: No module named 'notamodule'
```

KeyError is thrown when a key is not found.

```
> D1={'1':"aa", '2':"bb", '3':"cc"}
> D1['4']
Traceback (most recent call last):
File "<pyshell#15>", line 1, in <module>
D1['4']
KeyError: '4'
```

ImportError is thrown when a specified function can not be found.

```
>from math import cube
Traceback (most recent call last):
File "<pyshell#16>", line 1, in <module>
from math import cube
ImportError: cannot import name 'cube'
```

StopIteration is thrown when the next() function goes beyond the iterator items.

```
> it=iter([1,2,3])
> next(it)
1
> next(it)
2
```

```
> next(it)
3
> next(it)
Traceback (most recent call last):
File "<pyshell#23>", line 1, in <module>
next(it)
StopIteration
```

A small typing mistake can lead to an error in any programming language because we must follow the syntax rules while coding in any programming language.

This is the most common and basic error situation where you break any syntax rule like if you are working with Python 3.x version and you write the following code for printing any statement,

```
print "I love Python!"
```

```
SyntaxError: Missing parentheses in call to 'print'.
```

Because, Python 3 onwards the syntax for using the `print` statement has changed. Similarly if you forget to add colon(`:`) at the end of the `if` condition, you will get a **SyntaxError**:

```
if 7 > 5

   print("Yo Yo!")
```

```
SyntaxError: invalid syntax
```

EXPLANTION:

```
# This code has an intentional error. Do not type it directly;
# use it for reference to understand the error message below.
def print_message(day):
    messages = {
        'monday': 'Hello, world!',
        'tuesday': 'Today is Tuesday!',
        'wednesday': 'It is the middle of the week.',
        'thursday': 'Today is Donnerstag in German!',
        'friday': 'Last day of the week!',
        'saturday': 'Hooray for the weekend!',
        'sunday': 'Aw, the weekend is almost over.'
    }
    print(messages[day])

def print_friday_message():
    print_message('Friday')

print_friday_message()
```

ERROR

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-1-4be1945adbe2> in <module>()
     14     print_message('Friday')
     15
---> 16 print_friday_message()

<ipython-input-1-4be1945adbe2> in print_friday_message()
     12
     13 def print_friday_message():
---> 14     print_message('Friday')
     15
     16 print_friday_message()

<ipython-input-1-4be1945adbe2> in print_message(day)
      9         'sunday': 'Aw, the weekend is almost over.'
     10     }
---> 11     print(messages[day])
     12
     13 def print_friday_message():

KeyError: 'Friday'
```

SOLUTION:

- ☐ 3 levels
- ☐ `print_message`
- ☐ 11
- ☐ `KeyError`
- ☐ There isn't really a message; you're supposed to infer that `friday` is not a key in `messages`.