MCS 260 – Introduction to Computer Science – Fall 2020 – Emily Dumas

Week 2 Worksheet Solutions

(1) What is the difference between interactive mode (REPL) and script mode?

Answer:
In interactive mode, a single statement is entered at a prompt, evaluated immediately, and the result (if any) is printed.

In contrast, in script mode, all of the statements in a file are executed, and no output is produced unless explicitly called for (e.g. with `print()`).

(2) For each example below, answer two questions:
- What would be the output in the REPL?
- What would be the output if this were the only line of code in a script?

Answers:

code	REPL output	script output
`print("alpha")`	`alpha`	`alpha`
`"beta"`	`'beta'`	
`2**3 + 1`	`9`	
`int(19.9999)`	`19`	

(3) Some lines below have errors. Which ones? What is wrong? (If there is no error, what would the output be?)

(a)
```
print(I am learning a lot)
```

Answer:
This is an error. The words between `()` are interpreted as names, which are first of all unknown, and furthermore the syntax of names separated by spaces is not allowed in Python.

(b)
```
print("I am "learning" a lot")
```

Answer:
This is an error. The inner set of double quotes end and then restart the string literal, so that `learning` is interpreted as a name. This name is not known, and a name adjacent to a literal is not valid Python syntax.

(c)
```
print("I am \"learning\" a lot")
```

Answer:
This is valid and prints
```
I am "learning" a lot
```

(d)
```
print('I am learning a lot)
```

Answer:
This is an error. The string literal started by `'` does not end.

(e)
```
print("In the first week we had",3,"lectures")
```

Answer:
This is valid and prints
```
In the first week we had 3 lectures
```

(f)
```
print("In the first week we had" 3 "lectures")
```

Answer:
This is an error. An integer literal cannot be next to a string literal. If multiple values are to be printed with `print()`, they must be separated by commas.

(4) Determine the value of each expression (remember PEMDAS!)

(a)
```
2**5 + 2**3 - 0b101000
```

Answer:
0

(b)
```
2**3/3+1
```

Answer:
A float approximating $\frac{11}{3}$, specifically: 3.6666666666666665

(c)
```
1<<5 + (0xf0 & 0x0f)
```

Answer:
32

(d)
```
7 | 8
```

Answer:
15

(5) A variable `s` has been assigned the string value `"science!"`. Determine the value of each expression:

(a)
```
s[2]
```

Answer:
`i`

(b)
```
"it works" + s[7] * 3
```

Answer:
`it works!!!`

(c)
```
s[2]+s[5]+s[6]
```

Answer:
`ice`

Also, suppose that `i` is a variable that has a positive integer value.

(d) What are the possible values of `i` if `s[i]` is equal to `c`?

Answer:
1 and 5

Extensions: If you complete the problems above before the end of the discussion meeting, use the remaining time to explore the following. (These topics are not covered in quiz 2.)

- Can you get Python to produce an error due to a calculation with integers producing an integer that is too large?

  Answer:
  It is not easy, but it can be done. Unlike many other languages, Python ints do not have any built-in maximum size, and the interpreter will adjust the way integers are stored in memory so as to complete calculations that would exceed the maximum representable integer for a fixed number of bits. However, it is possible to request integer calculations that would exhaust a computer's limited resources (e.g. available memory).

- Can you get Python to produce an error due to a calculation with floats producing a float that is too large?

  Answer:
  Yes, a large int cannot be converted to a float:

```
>>> float(10**400)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
OverflowError: int too large to convert to float
```

- Can you get Python to produce an error due to a string being too long?

  Answer:
  Yes, but as with ints, it is not particularly easy. Strings of a few billion characters are supported on all platforms, unless the available memory becomes a limitation. The theoretical maximum size is platform-dependent, but on 64-bit platforms the maximum size is large enough that exhaustion of available memory will occurr before the true limit is reached.

- Another common bitwise operation is to simply flip all of the bits (0 becomes 1 and 1 becomes 0). This is called NOT. Can you see any problem with defining this operation on an integer?

  Answer:
  What would be the bitwise NOT of 0? In theory it should have binary digits "all 1s", but unless you limit the total number of bits, this does not make sense. However, there is no fixed number of bits that can represent all integers. Thus, if the domain is *integers*, then there is no bitwise NOT operation. If the domain is, say 16-bit integers, than such an operation is easily defined.

  In Python, ints have no fixed number of bits, and so there is no bitwise NOT operation.

- Does `+` on strings have a unit? That is, does there exist a string `z` so that `s+z` is always equal to `s` (where `s` is also a string)?

  Answer:
  The empty string `""` exists and is a unit for string addition.

Revision history:

- 2020-09-04 Initial release