Lab Manual 9 - System Ops, Exceptions

Lab 7 Answers

```python
# Optional Lab 7 Answers
def use_loot(belt, health_points):
    good_loot_options = ["Health Potion", "Leather Boots"]
    bad_loot_options = ["Poison Potion"]
    item_num = 0

    # Validate Arguments belt and health points
    if len(belt) <= 0 or len(belt) > 5:
        print("You cannot use loot with an empty belt, and you can't have more than 5 items on your belt")
    elif health_points <= 0 or health_points > 20:
        print("You cannot use loot if you're dead, and you can't have more than 20 health points")
    else:
        # Choose the loot to use

        # Lab 7 - Added a choice to use nothing from the belt
        print("Enter -1 : to use nothing")
        for item_option in belt:
            print("Enter " + str(item_num) + " : to use " + item_option)
            item_num += 1
```

```python
        # Lab 7 - Validate the item number entered
        i = 0
        input_invalid = True
        use_an_item = False
        while input_invalid and i in range(5):
            item_choice = input("Choose which item from your belt to use (Enter a Number):  ")
            if item_choice == "-1":
                print("Skipping Using Loot")
                input_invalid = False
            elif not item_choice.isnumeric():
                print("TRY AGAIN: Item number entered must be a number")
            else:
                item_choice = int(item_choice)
                item = belt.pop(item_choice)
                use_an_item = True
                input_invalid = False

        # If Item number was valid, Determine consequence of loot chosen
        if use_an_item and not input_invalid:
            if item in good_loot_options:
                health_points = min(20, (health_points + 2))
                print("You used " + item + " to increase your health to " + str(health_points))
            elif item in bad_loot_options:
                health_points = max(0, (health_points - 2))
                print("You used " + item + " to reduce your health to " + str(health_points))
            else:
                print("You used " + item + " but it's not helpful")
    # Lab 7 - Don't need to return belt list, because it was passed by reference above
    return health_points
```

System Operations

Platform

The platform module gives Access to underlying platform's identifying data
There are cross platform commands and platform-specific commands, specific for Windows, Java Machine, MacOS, Linux, etc.

`platform.machine()`

- **Return Type:** Returns the machine type, e.g. 'i386'. An empty string is returned if the value cannot be determined.

`platform.processor()`

- **Return Type:** Returns the (real) processor name, e.g. 'amdk6'.
- An empty string is returned if the value cannot be determined. Note that many platforms do not provide this information or simply return the same value as for machine(). NetBSD does this.

Socket

`socket.getdefaulttimeout()`

- Return the default timeout in seconds (float) for new socket objects. A value of None indicates that new socket objects have no timeout. When the socket module is first imported, the default is None.

`socket.setdefaulttimeout(timeout)`

- Set the default timeout in seconds (float) for new socket objects. When the socket module is first imported, the default is None. See settimeout() for possible values and their respective meanings.

The sys module

```
import sys
sys.exit()
```

The sys module gives you access to variables and functions used by the interpreter, or that work closely with the interpreter

Exiting the Program

Good for Development Phase: quit() and exit()

Good for Production Phase: sys.exit()

- It is good for Production Phase because it is always available on different computers and OSs .
- Remember: Production-phase is after the product has been released publicly for clientsThe os module

File Open and Close with OS

- Intended for low-level I/O
- File descriptors are **a low-level facility for working with files directly provided by the OS kernel**
- directory descriptors are **a low-level facility for working with directories directly provided by the OS kernel**

```
import os
```

Provides access to the operating system. If you just want to read or write a file, don't use this (even though you can) use `open()` instead.

```
print(os.name)
```

- **Parameter:** No parameter is required
- **Return Type:** Return the current name of the operating system
- **My Example returns:** POSIX stands for **Portable Operating System Interface**

```
print(os.getpid())
```

- **Parameter:** No parameter is required
- **Return Type:** Return the current process id.
- **My Example returns:** 68223

```
print(os.fork())
```

- **Parameter:** No parameter is required
- **Return Type:** This method returns an integer value representing the child's process id in the parent process while 0 in the child process.
- **My Example returns:** 68277

`fo = os.fdopen(fd, "w+")`

- This is an alias of the open() built-in function and accepts the same arguments.
    - The only difference is that the first argument of fdopen() must always be an integer.
- **Return Type:** Return an *open file object* connected to the *file descriptor fd*.

`os.close(fd)`

- Close the file descriptor fd.

`os.write(fd, str)`

- Write the bytestring in str to file descriptor fd.
- **Return Type:** Return the number of bytes actually written.

`os.read(fd, n)`

- Read at most n bytes from file descriptor fd.
- **Return Type:** Return a bytestring containing the bytes read. If the end of the file referred to by fd has been reached, an empty bytes object is returned.

Full Write and Read Example

```
# Lab 9 - Read and Write to a file object
fo = os.fdopen(fd, "w+")
print("File Object for fdpractice.txt: " + str(fd))
print("Current I/O pointer position :%d" % fo.tell())
fo.write("Some string to write to a file object")
fo.flush()
os.lseek(fd, 0, 0)
read_string = os.read(fd, 100)  # Should read Some string to write to
a file object
print(read_string)
fo.close()
print("--------------------------------")
```

`file_to_read, file_to_write = os.pipe()`

- You Sends the output (return value) of one system command as the input (command line argument) of another system command. For example, you might
- **Parameter:** none
- **Return Type:** returns a pair of file descriptors.
- You can see that it creates a pipe, or a connection between the output of `file_to_read` and the input of `file_to_write`

Exceptions

A programming statement can have correct syntax, and even correct logic, but still throw exceptions. Errors detected during execution are called *exceptions* and do not crash your program as long as you catch them with `except:`

Example

```
# Loop to get valid input for Hero Combat Strength
i = 0
while not input_valid and i in range(5):
    # Lab 9 - Add try catch block for combat strength
    # Lab 9 - Tells the program: I'm potentially expecting an error
    try:
    # Lab 9 - Try directly casting the input to an int
    # Lab 9 - If this throws an error, your except block will catch
it
        combat_strength = int(input("Enter your combat Strength (1-6):
"))
        # Validate input: Check if the string inputted
        if combat_strength not in range(1, 7):
            print("Enter a valid integer between 1 and 6 only")
            i = i + 1

        else:
            input_valid = True
    # Lab 9 - If error occurs, print custom message i
    # Make sure you're not still getting a red crash error
    except ValueError:
        print("Exception: Invalid input. Enter integer numbers for
Combat Strength")
```

Extra Credit Material

Full Pipe Example

```
# Lab 9 - Practicing System Operations
print(os.name)
print(os.getpid())
print(os.fork())
```

```
file_d_to_read, file_d_to_write = os.pipe()
process_id = os.fork()
if process_id:
    os.close(file_d_to_write)
    file_d_to_read = os.fdopen(file_d_to_read)
    print("Parent reading")
    str = file_d_to_read.read()
    print("text =", str)
    sys.exit(0)
else:
    # This is the child process
    os.close(file_d_to_read)
    file_d_to_write = os.fdopen(file_d_to_write, 'w')
    print("Child writing")
    file_d_to_write.write("Text written by child...")
    file_d_to_write.close()
    print("Child closing")

    sys.exit(0)
```