COL1000: Introduction to Programming

REVIEW CLASS

Subodh Sharma | Lec 12 | Aug 29

Reminders!

- Many students are still informing via email about missed labs -- refer to Prof. Subodh Kumar’s mail and use the form link to submit the medical certificate.
- **A medical certificate is required only for missed labs**. The policy for missed labs (formative assessments/exams) has already been shared via email.
- **No medical leave for lectures**. Students must maintain 75% attendance;

Recognising Common Errors

- Indentation Error: Not indented properly

```
#indention error
   x = 5
```

```
y", line 2
    x = 5
IndentationError: unexpected indent
```

- Name Error:

```
# Name Error

if(x > 10):
    pass
```

```
    if(x > 10):
       ^
NameError: name 'x' is not defined
```

Recognising Common Errors

- Value Error:

```
# Value Error

int('Hello')
```

```
    int('Hello')
    ~~~^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'Hello'
```

- Type Error:

```
x, y = 'Hello', 10
print(x+y)
```

```
    print(x+y)
          ~^~
TypeError: can only concatenate str (not "int") to str
```

Recognising Common Errors

- Attribute Error:

```
x = None
x.append(5)
```

```
    x.append(5)
    ^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'append'
```

- Index Error:

```
nums = [10, 20, 30]
print(nums[3])
```

```
    print(nums[3])
          ~~~~^^^
IndexError: list index out of range
```

Recognising Common Errors

- Key Error

```
Grades  = {"svs": 0}
print(Grades["rohit"])
```

```
    print(Grades["rohit"])
          ~~~~~~^^^^^^^^^
KeyError: 'rohit'
```

- DivByZero Error

```
x =0
y =3
print(y/(x*y))
```

```
    print(y/(x*y))
          ~^^~~~~
ZeroDivisionError: division by zero
```

Objects, Variables & References

Objects: Mutability

- **Mutable objects**: Objects whose contents can be changed without changing their identity (i.e id remains the same)
  - Examples: lists, dict, sets
- **Immutable objects:** Objects once created cannot be modified
  - Examples: int, float, bool, str, tuple

Objects

- Recall each object has a: **type**, **value**, **Id**
  - Use `type()`, `id()` to obtain type and Id information regarding the object

```
x = None
print(type(x), id(x))
```

```
<class 'NoneType'> 4460590136
```

  - Named variables for objects: **act just as labels**
  - Is vs == : Is operator checks the equality over Ids of its operands (`x is y` ≡ `id(x) == id(y)`); == is check of equality over values of its operands
  - `x += v` and `x = x+v` are different; former is in-place and doesn't create a new object while the latter does

References

- `a = b` (`b`'s reference is copied in to `a`)
- `a = b + 5` (fetch b's value through its reference and evaluate the expression)
- `l1 = ['5', 10, a]` and then `l2 = l1`
  - `print(id(l1) == id(l2)) -> True`
  - But lists are mutable. `l1.append("svs"); l1[3] = l1; l2 = l1 + l1`
    - Check now if the ids are the same?
- `x,y = v1, v1` (assign references from left to right)
- `x,y = [5, "svs"]` (unpacking of the object to constituents and then binding them to the references
- `l1 = ['5', 10, a]; l2 = l1; del l1` —> l1 name is deleted but the object is alive through reference l2