Week 1 Review

Python Built-in Functions

Another Python Data Type – Strings

Python Print and Input

Built-in Functions int, float, str

Programming = CT + Coding

High level languages: Python, etc.

Python primitive elements

Objects (type/value(s)/id)

Type – values and operations

Primitive elements can be combined

Expressions evaluate to a value

Assignment statements (are not expressions) associate name/object

Variables are expressions

Syntax, Runtime, and Logic Errors

Given:

```python
>>> x = 0
>>> x = x + 1
>>> x = x + 2
```

What would be the result?

```python
>>> type(x)
??
>>> x
??
```

```python
>>> x = 0
>>> x += 1  # same as x = x + 1
>>> x += 2  # same as x = x + 2
```

List all of the expressions

List all of the assignments

```python
>>> type(x)
<class 'int'>
>>> x
3
```
When something goes wrong:

```python
>>> temperature = 20
>>> temperature
20
>>> temperature + 20
40
>>> temperature ++ 20
# ok (logic?)
>>> temperature ** 2
# logic
```

"*Good [programming] comes from experience, and a lot of that comes from bad [programming]."* — [nod to] Will Rogers

Recall: Python Types

-- set of values
-- set of operations that can be applied to those values

Example:

- integers, floats
- +, -, round, pow, abs, print ...

Python Functions

A function names an operation.
Python Functions

A function names an operation.

>>> abs
<built-in function abs>

Calling a function:

>>> abs()

TypeError: abs() takes exactly one argument (0 given)

>>> abs(-7)
7

>>> y = 2.001
>>> round(y)
2
>>> pow(3, 2)
9

>>> y = 2.001
>>> round(y)
2
>>> pow(3, y)  # be careful
??
Python Functions

```python
>>> y = 2.001
>>> round(y)
2
>>> pow(3, round(y))
??
>>> y = round(y)
>>> pow(3, y)
??
```

For example,

```python
>>> y = 3.14159
>>> z = abs(round(y))
??
```

Another data type - Strings

```python
>>> 'Hello, world'
'Hello, world'
```

Another data type - Strings

```python
>>> '97403'
'97403'
```

```python
>>> type('97403')
<class 'str'>
```

Another data type - Strings

```python
>>> 97403
97403
```

```python
>>> Lincoln
NameError: name 'Lincoln' is not defined
```
• Objects
  – Value
  – Type (range of values and operations)
  – Id (unique identifier; memory location)

>>> x = 210
>>> y = '210'
>>> x
>>> y
>>> len(x)
>>> len(y)
>>> x / 3
>>> y / 3
>>> x + 3
>>> y + '3'
>>> x + '3'

Strings are sequences of characters

“hello, world”

0 1 2 3 4 5 6 7 8 9 10 11

>>> greeting = 'hello, world'
>>> len(greeting)
??

String operators
  + and * are overloaded operators

>>> 'try' + 'this'
??

>>> 'try this' * 4 + 'ok'
??

Strings are sequences of characters

“hello, world”

0 1 2 3 4 5 6 7 8 9 10 11

>>> greeting = "hello, world"
>>> len(greeting)
12

String operators
  + and * are overloaded operators

>>> 'try' + 'this'
'trythis'

>>> 'try this' * 4 + 'ok'
'try thisthry thisthry thisthry thisthry thisok'
Strings are sequences of characters

“hello, world”

0 1 2 3 4 5 6 7 8 9 10 11

>>> greeting[0]
>>> greeting[11]
>>> greeting[-1]

>>> greeting[len(greeting)]  # be careful
IndexError: string index out of range

>>> a = “ARMADILLO”
>>> c = “CHAMELEON”
>>> string0 = a[2:5]*2 + c[3:6] + c[7:len(c)]
>>> string0
??
Python sequential operator **for**

```python
for <var> in <sequence>:
    <block of code>
```

Python **Print** and **Input**

Another function that works with string args:

```python
>>> print('hello, world')
hello, world
```

```python
>>> greeting = 'hello,\tworld'
>>> len('greeting')
8
>>> greeting
'hello,\tworld'
>>> len(greeting)
12
>>> print(greeting)
hello, world
```
Python Print and `input`

```python
>>> name = input()
CIS 122

>>> name
'CIS122'

>>> print(name)
??
```

```python
>>> days_in_month = input('How many days? ')
How many days? 30

>>> days_in_month
'30'

>>> type(days_in_month)
<class 'str'>
```

```python
>>> days_in_month = input('How many days? ')
How many days? 30

>>> days_in_month
'30'

>>> day_before = days_in_month - 1 # careful
>>> total = .01 * (2 ** day_before)
```

```python
>>> days_in_month = input('How many days? ')
How many days? 30

>>> days_in_month
'30'

>>> day_before = days_in_month - 1
TypeError: unsupported operand type(s) for -: 'str' and 'int'

>>> total = .01 * (2 ** day_before)
```
Python Print and Input

```python
>>> days_in_month = int(input('How many days? '))
How many days? 30

>>> days_in_month = int(days_in_month)

>>> days_in_month
30

>>> type(days_in_month)
??
```

Python Print and Input – Good Bug!

What will be the result when this code is executed?

```python
>>> days_worked = int(input('How many days? '))
How many days? 20

>>> pay_check = days_worked * 120

>>> days_worked
20

>>> type(days_worked)
??

>>> days_worked = str(days_worked)

>>> days_worked
'20'

>>> days_worked = float(days_worked)
```

Solving a problem computationally:

Start with the project specification – be sure you understand what is being asked:

- What is the desired outcome/result?
- What information is needed?
- Is the information available (check spec)?
- Solve a simple example of the problem, starting from available information and stopping when the result is achieved.
- Write down the problem-solving steps/algorithm. The more specific you can be, the better.

Moving from the algorithm to Python code:

- What variables will you need? For the given information? For the result?
- What types are the variables? What Python operations are available (operators, built-in functions)?
- Or, conversely, what operations are needed? What types of data/variables are needed?
- Turn the algorithm steps into Python code.
- Report the result (e.g., print).