Basic Formulas in Python

ENGR 298: Engineering Analysis and Decision Making with Python

Equations: a very basic calculations

- Consider the equation below that determines interest on bank deposits. What is the **data**? For each datum, what is the variable name? What are the units for each variable?

$$A = P(1 + (r/100))^n,$$

where P is the initial deposit (the *principal*), r is the yearly interest rate given in percent, n is the number of years, and A is the final amount.

A program is a set of instructions for **manipulating**, **moving**, and **making decisions** on **data** in your computer.

Equations: a very basic calculations

$$A = P(1 + (r/100))^n,$$

where P is the initial deposit (the *principal*), r is the yearly interest rate given in percent, n is the number of years, and A is the final amount.

Data	Variables Name	Units	Integer / Float

Equations: a very basic calculations

$$A = P(1 + (r/100))^n,$$

where P is the initial deposit (the *principal*), r is the yearly interest rate given in percent, n is the number of years, and A is the final amount.

Data	Variables Name	Units	Integer / Float
Principal	P	USD (could be any currency)	Two decimal point float
Interest rate	r	unit less	Floating point
Number of years	n	Years	Either; 1 year or 2.25 years
Final amount	A	USD (could be any currency)	Two decimal point float

Equations: a very basic calculations

- What **manipulations** are being applied to each variable? What is the order of these operations? Is any data being **moved/stored**?

$$A = P(1 + (r/100))^n$$

Consider all operations that may be implied by how the equation is written….

A program is a set of instructions for **manipulating**, **moving**, and **making decisions** on **data** in your computer.

Equations: a very basic calculations

- What **manipulations** are being applied to each variable? What is the order of these operations? Is any data being **moved/stored**?

$$A = P(1 + (r/100))^n$$

Operand A	Operator	Operand B
r	Division (/)	100
1	Addition (+)	(r/100)
(1+(r/100))	Power (^)	n
P	Multiplication (*)	(1+(r/100)^n

A program is a set of instructions for **manipulating**, **moving**, and **making decisions** on **data** in your computer.

Equations: a very basic calculations

- Are any **decisions** required to render the correct result?

$$A = P(1 + (r/100))^n,$$

where P is the initial deposit (the *principal*), r is the yearly interest rate given in percent, n is the number of years, and A is the final amount.

A program is a set of instructions for **manipulating**, **moving**, and **making decisions** on **data** in your computer.

Calculations in Python

- Basic arithmetic operations (+−/∗ ^) are directly supported in the Python language along with simple number types (*int*, *float*, and *complex*) Each variable ‘type’ is dynamically determined by Python so it is important that values are declared appropriately.

- Operators will return results (*int* or *floats*) based upon the variable ‘types’. Long floating point values should be retained throughout the calculation and then truncated at the end (avoid data loss).

- Crucial to utilize () to respect order of operations. Do not assume the program will “know what you mean”.

3.1.1. Numbers

The interpreter acts as a simple calculator: you can type an expression at it and it will write the value. Expression syntax is straightforward: the operators `+`, `-`, `*` and `/` work just like in most other languages (for example, Pascal or C); parentheses (`()`) can be used for grouping. For example:

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # division always returns a floating point number
1.6
```

The integer numbers (e.g. `2`, `4`, `20`) have type `int`, the ones with a fractional part (e.g. `5.0`, `1.6`) have type `float`. We will see more about numeric types later in the tutorial.

Division (`/`) always returns a float. To do floor division and get an integer result (discarding any fractional result) you can use the `//` operator; to calculate the remainder you can use `%`:

Operation	Result	Notes	Full documentation
`x + y`	sum of *x* and *y*		
`x - y`	difference of *x* and *y*		
`x * y`	product of *x* and *y*		
`x / y`	quotient of *x* and *y*		
`x // y`	floored quotient of *x* and *y*	(1)	
`x % y`	remainder of `x / y`	(2)	
`-x`	*x* negated		
`+x`	*x* unchanged		
`abs(x)`	absolute value or magnitude of *x*		`abs()`
`int(x)`	*x* converted to integer	(3)(6)	`int()`
`float(x)`	*x* converted to floating point	(4)(6)	`float()`
`complex(re, im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.	(6)	`complex()`
`c.conjugate()`	conjugate of the complex number *c*		
`divmod(x, y)`	the pair `(x // y, x % y)`	(2)	`divmod()`
`pow(x, y)`	*x* to the power *y*	(5)	`pow()`
`x ** y`	*x* to the power *y*	(5)	

https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex

Implementing Equations in Python

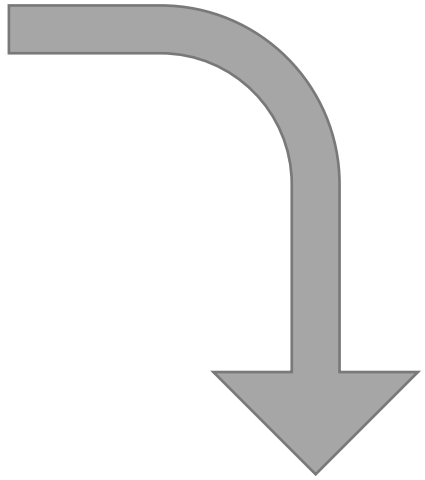
- Take time to clearly declare and COMMENT each variable
- When providing values for each variable, consider what are reasonable values? E.g. Should *rate* be 0.01 or 1?
- Utilize () to ensure order of operations is respected.
- Place print() statements to make the program readable (to you and others). Acts as documentation and sanity check.

```python
principal = 1000  # initial amount to be deposited
rate = 1  # interest rate applied to deposit (will be divided by 100)
n = 10  # number of years to compound deposit

# must cast n as 'string' so it can be printed in print()
print("Initial principal is $" + str(principal))
print("Interest rate is " + str(rate / 100))
print("Hold for " + str(n) + " years")

# calculate the eventual result
final = principal * (1 + (rate / 100)) ** n

# fancy print the output with two decimal places for floating number
print(f"Final value after is ${final:.2f}")
```



```
Initial principal is $1000
Interest rate is 0.01
Hold for 10 years
Final value after is $1104.62
```

Cast variables to string so can be passed to print()

```python
principal = 1000  # initial amount to be deposited
rate = 1  # interest rate applied to deposit (will be divided by 100)
n = 10  # number of years to compound deposit

# must cast n as 'string' so it can be printed in print()
print("Initial principal is $" + str(principal))
print("Interest rate is " + str(rate / 100))
print("Hold for " + str(n) + " years")

# calculate the eventual result
final = principal * (1 + (rate / 100)) ** n

# fancy print the output with two decimal places for floating number
print(f"Final value after is ${final:.2f}")
```

Truncate variable *final* to 2 decimal places

https://github.com/jforsyth/ENGR298-sp2022/blob/main/lecture-examples/week2-formulas.py

What about more complex operations…

- The power operation was implemented with ** but what if that operation was not supported? Or, what if more complex operations were required?

- Python utilizes *modules* to store and manage large sets of functions. *Modules* may “come with” Python or need to be installed via package manager (pip, brew, apt-get)

- Keyword “import” will bring in a module. Can import whole package or just elements of each package.

`math` — Mathematical functions

https://docs.python.org/3/library/math.html

This module provides access to the mathematical functions defined by the C standard.

These functions cannot be used with complex numbers; use the functions of the same name from the `cmath` module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

The following functions are provided by this module. Except when explicitly noted otherwise, all return values are floats.

`math.`**`pow`**`(x, y)`

Return `x` raised to the power `y`. Exceptional cases follow Annex 'F' of the C99 standard as far as possible. In particular, `pow(1.0, x)` and `pow(x, 0.0)` always return `1.0`, even when `x` is a zero or a NaN. If both `x` and `y` are finite, `x` is negative, and `y` is not an integer then `pow(x, y)` is undefined, and raises `ValueError`.

Unlike the built-in `**` operator, `math.pow()` converts both its arguments to type `float`. Use `**` or the built-in `pow()` function for computing exact integer powers.

Comparison of import methods and using pow() for 3^2

```
# import the math library
import math

math.pow(3, 2)
```

- *Style imports whole math module so much specify which function in math module.*

```
# import pow() from math library
from math import pow

pow(3, 2)
```

- *Imports specific function from math module. Do not need to specify “math.” as pow() is already known/defined*

```python
# import math to utilize pow()
import math

principal = 1000  # initial amount to be deposited
rate = 1  # interest rate applied to deposit (will be divided by 100)
n = 10  # number of years to compound deposit

# must cast n as 'string' so it can be printed in print()
print("Initial principal is $" + str(principal))
print("Interest rate is " + str(rate / 100))
print("Hold for " + str(n) + " years")

# calculate the eventual result, use pow() from math library
final = principal * math.pow((1 + (rate / 100)), n)

# fancy print the output with two decimal places for floating number
print(f"Final value after is ${final:.2f}")
```

https://github.com/jforsyth/ENGR298-sp2022/blob/main/lecture-examples/week2-math-package.py

Create your own Python scripts to answer the following questions. Submit to Canvas.

- **Musk Money**: Recently, Elon Musk sold $1.02 Billion in TSLA shares. If he placed the entire sale in US Treasury Bonds, what would the sale be worth after 10 years? 20 years? Ignore capital gains taxes and highly paid accountants.
  - **Food for thought**: If Musk placed the entire sale in US Treasury Bonds and lived off the interest alone, what would be his annual income?
  - **Food for thought** : Assuming university endowment contributions ≥$1M earn 5% year, how large a gift to JMU is required to fund 1x student scholarship perpetually?

- **Your Money**: You may wish to retire in ~40 years. In planning for that event you open an retirement account. If you place $6k in the account every year, and in general the market returns 7% per year, what will be the final value of your account? Implement this as a compound interest calculation in a new Python script.
  - **Food for thought**: You have now spent your whole life saving and wish to retire in comfort just like Elon Musk. If you placed your life savings from above in a treasury bond, what would be your annual income? Do you believe this is enough to live on? Ignore taxes and inflation.