Class 2: Ideas, Tools, Functions

Note that the function is probably the most powerful tool in programming.

1. **Twenty questions Class Project Progress – Iteration 3**

It's a marathon (short 😊), not a sprint.

Iteration 3: Say if guess is greater, less or equal

1. **Print with optional prefix – pprint.py**

It is often useful to have a "print" function which acts like the traditional python print function but does something "special" for us, for example, precede the printed text with an optional prefix. Write and test a function:

pprint(arg, end=None, sep=None, prefix=None)

pprint acts like the standard print function**, with one arg**, but precedes the standard output with the value of prefix, if present.

Sample **pprint.py** output:

>>>

= RESTART: C:/Users/raysm/workspace/python/IntroductionToProgramming/homework/Class\_2\_\_Ideas\_Tools\_Functions/pprint/pprint.py

Test pprint

Testing no prefix: arg: Our String

Our String

Testing with prefix: arg: Our String prefix: PREFIX:

PREFIX:Our String

Testing with variable prefix:

Increasing prefix

=Increasing prefix

==Increasing prefix

===Increasing prefix

====Increasing prefix

=====Increasing prefix

======Increasing prefix

>>>

1. **Simple polygons, using turtle – polygons.py**

Given the module turtle has rather basic operations such forward, right, one might like to add more elaborate figures to facilitate the creation of larger drawings. In this exercise, create and test the following function to create a rectangle:

rectangle(x=0, y=0, height=100, width=100, color="red")

which draws a red rectangle, starting at the coordinates 0,0 with the height of 100 and the width of 100.

Sample polygons.py output:

>>>

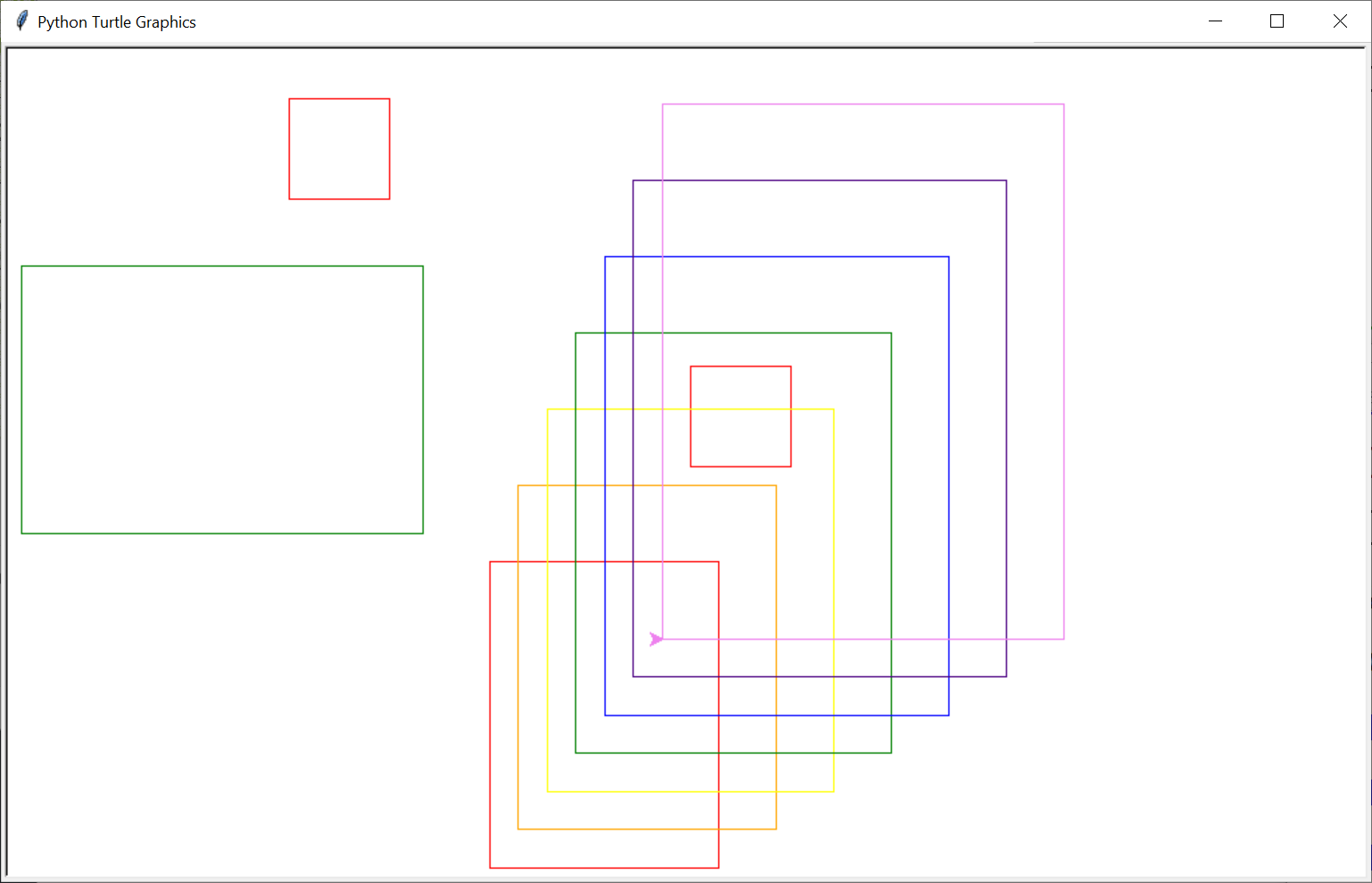
= RESTART: C:\Users\raysm\workspace\python\IntroductionToProgramming\homework\Class\_2\_\_Ideas\_Tools\_Functions\polygons\polygons.py

Simple Test - default rectangle

Simple Test - default size

Simple Test - one rectangle

Testing rectangle

Sample polygons.py Turtle Graphics screen: 

1. **More print keyword exercise - …homework/…/factors/…**

Redo / extend factors work:

* 1. **Exercise 2: – Print factors – factors\_v1.py**
  2. **Omit 1, and number itself as factors - factors\_v2.py**
  3. **Only print numbers with at least one factor other than 1 or itself - factors\_v3.py**
  4. **Ask from, to - factors\_from\_to.py**

1. **Salary Calculator**

Sometime it may be nice to convert salary between yearly, monthly, weekly, and hourly. This program should convert salary in one pay period to the equivalent pay in all pay periods.

To protect against user typo on numeric input, you might use the following:

while…

try

amt = float(inp)

except ValueError:

print(f"input:{inp} is not a valid number")

print("please try again")

continue

Salary Calculator

Loop

Ask salary pay period: Year, Month, Week, Hour

Ask salary, for this pay period:

Print salary yearly, monthly, weekly, hourly pay rates.

Sample **my\_salary.py** output:

>>>

= RESTART: C:\Users\raysm\workspace\python\IntroductionToProgramming\homework\Class\_2\_\_Ideas\_Tools\_Functions\budget\_calculator\my\_salary.py

Salary Calculator

supporting pay periods of:

year

month

week

hour

Enter BYE to quit

Enter pay period[year]: year

Enter pay amount per year: 65000

Salary: yr: 65,000.00 mth: 5,416.67 wk: 1,300.00 hr: 32.50

Enter pay period[year]: month

Enter pay amount per month: 3000

Salary: yr: 36,000.00 mth: 3,000.00 wk: 720.00 hr: 18.00

Enter pay period[month]: 2000

Sorry - we don't currently support pay periods of 2000

Enter pay period[month]: week

Enter pay amount per week: 1000

Salary: yr: 50,000.00 mth: 4,166.67 wk: 1,000.00 hr: 25.00

Enter pay period[week]: bye

Good Bye

>>>

1. **Budget Calculator (Not so easy)**

Sometimes it may be useful to see how budget goals (e.g., save 200/year reflect on other time periods). This exercise is to build a program which takes an amount / percentage per time period (%, year, month, week) reflects on the other time periods. **Hint**: Start with my\_salary.py

* 1. Ask salary pay period: Year, Month, Week
  2. Ask salary, per pay period:
  3. Loop
     1. Ask budget(save) period: Percent, Year, Month, Week
     2. Ask budget(save) amount: per period or percentage
     3. Print percentage, yearly amount, monthly amount, weekly amount

Sample **my\_budget.py** Output:

>>>

= RESTART: C:/Users/raysm/workspace/python/IntroductionToProgramming/homework/Class\_2\_\_Ideas\_Tools\_Functions/budget\_calculator/my\_budget.py

Budget Calculator

supporting pay periods of:

year

month

week

hour

Enter BYE to quit level

Enter pay period[year]:

Enter pay amount per year:

Assuming amt: 1000.

Salary: yr: 1,000.00 mth: 83.33 wk: 20.00 hr: 0.50

Enter Budget period or %[%]

Assuming: %

Enter Budget percent:[10]:

Budget: 10.0% yr: 100.00 mth: 8.33 wk: 2.00 hr: 0.05

Enter Budget period or %[%]month

Enter monthly Budget[0]:8.333

Budget: 10.0% yr: 100.00 mth: 8.33 wk: 2.00 hr: 0.05

Enter Budget period or %[month]hour

Enter hourly Budget[8.333]:.1

Budget: 20.0% yr: 200.00 mth: 16.67 wk: 4.00 hr: 0.10

Enter Budget period or %[hour]%

Enter Budget percent:[10]:20

Budget: 20.0% yr: 200.00 mth: 16.67 wk: 4.00 hr: 0.10

Enter Budget period or %[%]bye

Enter pay period[year]: bye

Good Bye

>>>

1. **Multiplication Table**

In my early school days, when attempting to learn and reinforce my arithmetic skills I would sometimes create multiplication tables which would display the product of two numbers from 1 to 12, e.g., 7\*8 giving 56. The table would have the number of one operand across the top and the other operand displayed down the left side. This exercise will be the creation of such a table.

The following is a run output for a simple implementation:

>>>

= RESTART: C:\Users\raysm\workspace\python\IntroductionToProgramming\exercises\times\_tables\times\_tables\_simple\_for.py

Enter times table length: 6

1 2 3 4 5 6

2 4 6 8 10 12

3 6 9 12 15 18

4 8 12 16 20 24

5 10 15 20 25 30

6 12 18 24 30 36

>>>

= RESTART: C

One might start with this goal and add:

* 1. Formatting the products to be right aligned
  2. Legends on top and left which display the operands
  3. Support the specification of maximum and minimum for each operand.

The following is a run output with the above extensions:

= RESTART: C:/Users/raysm/workspace/python/IntroductionToProgramming/exercises/mult\_table/mult\_table\_formated\_wleg.py

nmin[1]: 999999

nmax[1000004]: 1000001

mmin[999999]:

mmax[1000001]:

table 999999 to 1000001 by 999999 to 1000001

999999 1000000 1000001

+------------- ------------- -------------

999999| 999998000001 999999000000 999999999999

1000000| 999999000000 1000000000000 1000001000000

1000001| 999999999999 1000001000000 1000002000001