Writing functions

Pyret fun pen-cost(num-pens :: Number, msg :: String) -> Number: doc: "Computes the cost of order num-pens number of pens with msg on them" num-pens * (0.25 + (string-length(msg) * 0.02)) # where-examples go here end Python

Notes (how to run a file, how to interact with a file, calling functions):

If-expressions

Pyret

```
fun add-shipping-cost(order-price :: Number) -> Number:
  doc: ```Adds correct shipping cost to order-price.```
  if order-price <= 0:</pre>
   0
  else if order-price <= 12:</pre>
    order-price + 4
  else:
   order-price + 6
  end
# where-examples go here
end
Python
```

Testing

Notes on setting up testing:

```
Pyret
where:
    pen-cost(4, "") is 4 * 0.25
    pen-cost(1, "hi") is 0.29
    pen-cost(2, "smile") is 0.70
end

where:
    add-shipping-cost(0) is 0
    add-shipping-cost(2) is 6
    add-shipping-cost(12) is 16
    add-shipping-cost(20) is 26
end

Python
```

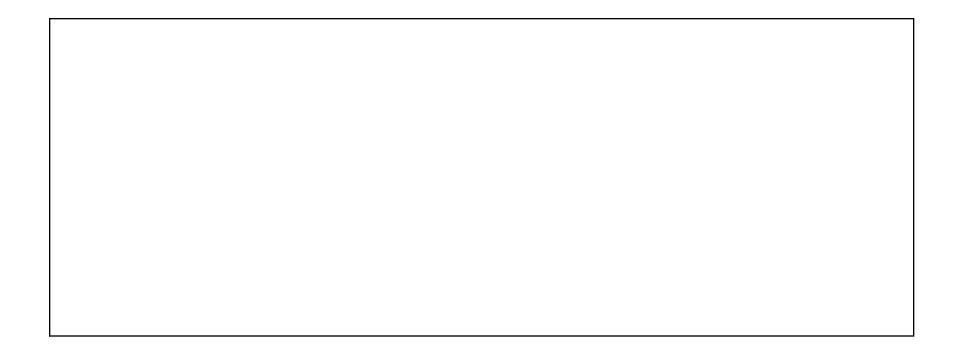
```
# fun pen-cost(num-pens :: Number, msg :: String) -> Number:
# doc: "Computes the cost of order num-pens number of pens with msg on them"
# num-pens * (0.25 + (string-length(msg) * 0.02))
```

```
# where:
  pen-cost(4, "") is 4 * 0.25
# pen-cost(1, "hi") is 0.29
# pen-cost(2, "smile") is 0.70
# end
# fun add-shipping-cost(order-price :: Number) -> Number:
   doc: ```Adds correct shipping cost to order-price
        If the order is no more than $12, add $4 shipping
        Otherwise, add $6 shipping
        Produce 0 if order-price is less than or equal to 0. ```
   if order-price <= 0:
     0
   else if order-price <= 12:</pre>
     order-price + 4
   else:
     order-price + 6
   end
# where:
   add-shipping-cost(0) is 0
  add-shipping-cost(2) is 6
   add-shipping-cost(12) is 16
   add-shipping-cost(20) is 26
# end
```

	Interacting	Using result in subsequent computation	Running
<pre>def add1_a(n): print(n + 1)</pre>			
<pre>def add1_b(n): return n + 1</pre>			
<pre>def add1_c(n): n + 1</pre>			

if-expressions and return

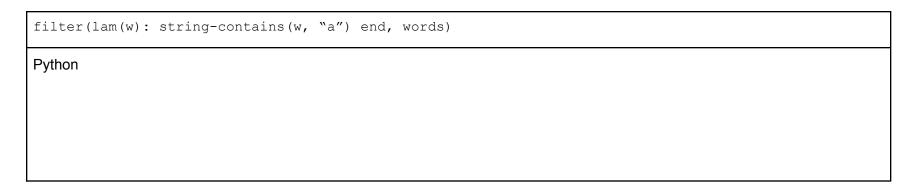
```
Pyret
fun add-1-if-pos(n :: Number) -> Number:
    if n > 0:
        n + 1
    else:
        n
    end
end
Python
```



Lists/Strings

```
Pyret

words = [list: "cat", "potato", "yarn", "noodle", "fern"]
length(words)
member(words, "cat") # true
member(words, "dog") # false
string-contains("cat", "at")
```



Summing a list

```
Order of computation in Pyret
```

```
num-lst = [list: 3, 7, 5]
sum(num-lst) will be
3 + sum([list: 7, 5])
    7 + sum([list: 5])
    5 + sum([list: ])
    0
```

Order of computation in Python

Code in Python