How Much? Write a function, `shipping`, which determines the shipping cost of an order. Shipping is $0.05 per pound but for shipments 100 pounds or more, subtract $1.50.

```python
def shipping(w):
    '''
    Given weight w, a positive number, return shipping cost ($0.05/pound, $1.50 discount for weights of 100 pounds or more).
    >>> shipping(10)
    Shipping cost is $0.50.
    >>> shipping(100)
    Shipping cost is $3.50.
    '''
    cost = w * .05
    # what about when w >= 100?
    return...
```
Python if statements

if is a Python keyword
if statements affect the program's flow of control
one path (or another) is selected for execution, depending on a condition(s)

Boolean Expressions/Conditional Statements

if <boolean expression>:
    <block of code> may execute or not
<next Python statement>

def shipping(weight):
    
    Return cost of shipping freight of weight.
    Cost is $.05 per unit, with $1.50 subtracted if weight is 100 units or more.
    
    >>> shipping(10)
    Shipping cost is $.50.
    >>> shipping(100)
    Shipping cost is $3.50.
    
    cost = weight * .05
    if weight >= 100:
        # when condition "weight >= 100" is True
        # then this block of code is executed.
        # (otherwise, it is not executed)
        cost = cost - 1.50
        # outside of if-block!
    print("Shipping cost is ${cost:.2f}.")
    return

    if 4 > 5:
        print('surprise')
        depending on the value of the boolean expression
Boolean Expressions/Conditional Statements

If `<boolean expression>`:
  `<block of code>`  may execute or not
  `<next Python statement>`

```python
>>> if 4 > 5:
    print('surprise')
True
>>> 4 > 5
False
(there are only 2 Boolean values)
```

Python if statements

```python
>>> weight = 100
>>> cost = weight *.05
>>> weight >= 100   # a Boolean expression
True  # returns a Boolean value
>>> if weight >= 100:  # if expression evaluates to True
    cost = cost - 1.50  # the code in the block runs
>>> cost
??
```

Boolean is another of Python’s built-in data types:

- Integer
- Floating point
- String
- Function
- **Boolean** values: True, False
  - logical operators: not, and, or
  - relational operators: < > <= >= == !=

Boolean expressions: relational and logical operators return Boolean values

```
a < b         not a < b
a <= b       a <= b and c >= d
a > b         a <= b or c >= d
a <= b
a == b       # use Boolean values/expressions (only)
a != b       # with logical operators
```
Order of Operations

- arithmetic – **, unary +, *, /, //, %, +, -
- relational – all at same level of precedence - <, <=, >, >=, ==, !=
- boolean – not, and, or

for same level of precedence – left to right

Relational operators return Boolean values

```python
>>> 6 < 10
True

>>> 6 != 10
True

>>> 6 == 10
False

>>> d = 99
True

>>> d < 100
True

>>> d <= 99
True
```

Boolean operators

```python
>>> not (d == 99)
False

>>> (90 < d) and (d < 100)
True

>>> 90 < d < 100
True

>>> x = 99
True

>>> y = 100
False

>>> (x > 99) or (y > 99)
False

>>> (x > 100) or (y > 100)
False
```

Truth Tables

<table>
<thead>
<tr>
<th></th>
<th>not</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
Truth Tables

and  or (inclusive or)
T  T  T  T
F  T  F  T
T  F  F  T
F  F  F  F

cold  windy  not cold and windy
T  T  T
T  F  F
F  T  F
F  F  F

cold  windy  (not cold) and windy  not(c and w)
T  T  (not T) and T  not(T and T)
F  T  F and T  not(T)
F  F  F and F  not(F)
T  F  (not T) and F  not(T and F)
F  T  F and F  not(F)

Boolean operators

d = 99
>>> not (d == 99)
False
>>> (90 < d) and (d < 100)
True
>>> 90 < d < 100  # nifty Python
True
>>> x = 99
>>> y = 100
>>> (x >= 99) or (y >= 100)
True
>>> (x > 100) or (y > 100)
False
Python if (selection/conditional) statements

>>> a = 5
>>> b = 3
>>> if a > b:
    c = 10
>>> c
??
Python if (selection/conditional) statements

```python
>>> a = 5
>>> b = 10
>>> if a > b:
    c = 10
# be careful!
```

31

Python if statements

```python
>>> a = 5
>>> b = 10
>>> if a > b:
    c = 10  # fork-in-the-road 1
d = True
else:
    c = 15  # fork-in-the-road 2
d = False
>>> c  # can take only 1 of these paths!
>>> d
```

32

Python if statements

```python
>>> a = 5
>>> b = 3
>>> if a > b:
    c = 10  # fork-in-the-road 1
d = True
else:
    c = 15  # fork-in-the-road 2
d = False
>>> c  # can take only 1 of these paths!
>>> d
```

33

Python if statements

```python
>>> a = 5
>>> b = 3
>>> if a > b:
    c = 10  # fork-in-the-road 1
d = True
elif a < b:
    c = 15  # fork-in-the-road 2
d = False
elif a == b:
    c = 20  # fork-in-the-road 3
d = False
else:
    print('something is wrong!')
>>> c  # can take only 1 of these paths!
```

34

Python if statements

```python
>>> a = 5
>>> b = 3
>>> if a > b:
    c = 10  # fork-in-the-road 1
e1f a < b:
    c = 15  # fork-in-the-road 2
e1f a == b:
    c = 20  # fork-in-the-road 3
e1se:
    print('something is wrong!')
>>> c  # can take only 1 of these paths!
```

35

Python if statements

```python
>>> a = 5
>>> b = 3
>>> if a > b:
    c = 10  # fork-in-the-road 1
e1f a < b:
    c = 15  # fork-in-the-road 2
e1f a == b:
    c = 20  # fork-in-the-road 3
e1se:
    print('something is wrong!')
>>> c  # can take only 1 of these paths!
```

36
CIS 122 Intro to Programming and Computational Problem Solving

```python
def q1(score):
    gradepoint = 0
    if score >= 90:
        gradepoint = 4
        Be careful!
    elif score >= 80:
        gradepoint = 3
    elif score >= 70:
        gradepoint = 2
    elif score >= 60:
        gradepoint = 1
    print(gradepoint)
    return
```

What is the result of executing
```
>>> q1(90)
```
```
gradepoint = 0
if score >= 90:
    gradepoint = 4
    Be careful!
else:
    gradepoint = 3
    >>> q1(50)
```
```
gradepoint = 2
else:
    gradepoint = 1
    >>> q1(60)
```
```
gradepoint = 1
print(gradepoint)
return
```

Now what is the result of executing
```
>>> q1(90)
```
```
gradepoint = 0
if score >= 90:
    gradepoint = 4
    Be careful!
elif score >= 80:
    gradepoint = 3
    >>> q1(50)
```
```
gradepoint = 2
elif score >= 70:
    gradepoint = 2
    >>> q1(60)
```
```
gradepoint = 1
print(gradepoint)
return
```

CIS 122 Intro to Programming and Computational Problem Solving

Python if statements
```
>>> a = 5
>>> a = 3
>>> a = 3
>>> b = 3
>>> b = 5
>>> b = 3
```
```
>>> if a > b:
    c = 10
    # fork-in-the-road 1
elif a < b:
    c = 15
    # fork-in-the-road 2
else:
    c = 20
    # fork-in-the-road 3
```
```
>>> c
# can take only 1 of these paths!
```

✓ Python Visualizer demo
✓ Midterm 1 comments
  • Project comments
  • Python conditional statements, cont’d ✓
  • Boolean data type and operations, cont’d ✓
  • Python built-in function input
  • Python type conversion functions
def temp_alert(temp):
    '''
    print information about the temperature'''
    if temp >= 90:
        print('hot')
        return
    elif temp >= 80:
        print('very warm')
        return
    elif temp >= 70:
        print('warm')
        return
    elif temp >= 60:
        print('cool')
        return

# What is the result of executing the following code:
>>> temp_alert(90)
be careful!

# What is the result of executing the following code:
>>> temp_alert(80)
be careful!

Python Input
>>> name = input('What is your name? ')  # helpful!
What is your name? CIS 122
>>> name
'CIS 122'
>>> type(name)
<class 'str'>

Python Input
>>> days_in_month = input('How many days? ')  
How many days? 31
>>> days_in_month
'31'
>>> type(days_in_month)
<class 'str'>
**Python Input**

```python
>>> days_in_month = input('How many days?
How many days? 31
>>> penultimate_day = days_in_month - 1
```

```
TypeError: unsupported operand type(s) for -: 'str' and 'int'
```

**Python Input**

```python
>>> days_in_month = input('How many days? ')
How many days? 31
>>> days_in_month = int(days_in_month)
>>> days_in_month
31
>>> type(days_in_month)
<class 'int'>
```

**Boolean expressions and values**

```python
answer = input('please respond y or n: ')
please respond y or n: y
if answer == 'y':
    some_var = True
else:
    some_var = False
some_var = (answer == 'y')
```

---

**REFERENCES**

- A structured approach to computational problem solving
- Python toolkit so far
- Programming concepts so far
- CIS 122 is a community of learners ...
- What can you expect in CIS 122?
- Suggested time management for CIS 122
- CIS 122 learning outcomes
A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

What can you expect in CIS 122?

☑ Weekly projects and exercises to support learning of computational problem solving in a variety of areas

Supported by

☑ Class – large group – big picture concepts, exercises, Q/A
☑ Lab – small group – exercises, computers/whiteboards, Q/A
☑ Class notes (posted), project solutions (posted), text readings – review, practice, explore
☑ Help hours – daily drop-in help, tutorials, code reviews
☑ Comprehensive assessments – 2 midterm + 1 final “demo”
☑ Individual feedback on weekly projects

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

CIS 122 is a community of learners where

• Everyone is welcome
• Everyone is respected
• We value intellectual challenges and deliberate practice in pursuit of new knowledge and skills
• We support and encourage each other
• We celebrate our own and each other’s accomplishments

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

CIS 122

Python toolkit so far

numeric data types (int, float) and operations (e.g., +, -, /, %, round, pow, abs)
string data types and operations (e.g., “Hello World”)
boolean data types and operations (e.g., True, False)
variables (identifiers) – associate a name with an object
expressions – combination of values, variables, and operations; a value or variable all by itself is an expression
reserved words: def, docstring, return
more built-in functions – print, type, help, min, max, input, exit

Python syntax (for loops)

functions change top-to-bottom control flow in Python programs – function is executed when it is called

Conditional statements affect Python flow of control. Conditional statements use boolean values.

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

CIS 122

Python toolkit so far

numeric data types (int, float) and operations (e.g., +, -, /, %, round, pow, abs)
string data types and operations (e.g., “Hello World”)
boolean data types and operations (e.g., True, False)
variables (identifiers) – associate a name with an object
expressions – combination of values, variables, and operations; a value or variable all by itself is an expression
reserved words: def, docstring, return
more built-in functions – print, type, help, min, max, input, exit

Python syntax (for loops)

functions change top-to-bottom control flow in Python programs – function is executed when it is called

Conditional statements affect Python flow of control. Conditional statements use boolean values.
Welcome to CIS 122
Intro to Programming and Computational Problem Solving

CIS 122 Learning Outcomes

• use a computational problem-solving approach to generate computer solutions (programs) to a variety of problems
• write well-structured, well-documented programs using the Python programming language
• be prepared to continue to study programming or computer science on your own or in other courses