CS 161: Introduction to Programming and Problem-solving

Warren Harrison

Program correctness: More complex control flow

PORTLAND STATE UNIVERSITY

Choice

- Depending on the outcome of a condition, you execute one set of instructions or another
- This is how computer programs appear to have “intelligence”

PORTLAND STATE UNIVERSITY

The if Statement

Syntax:

```
if(condition):
    code block
elif (condition):
    code block
else:
    code block
```

3

PORTLAND STATE UNIVERSITY

An Example

- To Compute Gross Pay:
  - if hours worked is more than 40, sum the hourly rate multiplied by 40 with the number of hours worked over 40 multiplied by 1.5 times the hourly rate
  - If hours worked is 40 or less, multiply the number of hours worked by the hourly rate
- Call the result gPay

4

PORTLAND STATE UNIVERSITY

Computing Gross Pay

```
hours = int(input("hours? "))
rate = float(input("rate? "))
if(hours > 40):
    gPay = 40 * rate
    gPay = gPay + (hours-40) * (rate*1.5)
else:
    gPay = hours * rate
print("Gross Pay: ",gPay)
```

5

PORTLAND STATE UNIVERSITY

A More Complicated Example to Compute Gross Pay

- if hours worked is more than 80, sum the hourly rate multiplied by 40, with the hourly rate times 1.5 multiplied by 40, and the hours worked over 80 multiplied by the hourly rate times * 2
- if hours worked is more than 40 but less than 80, sum the hourly rate multiplied by 40 with the number of hours worked over 40 multiplied by 1.5 times the hourly rate
- If hours worked is 40 or less, multiply the number of hours worked by the hourly rate
- Call the result gPay

6

PORTLAND STATE UNIVERSITY

Computing a More Complicated Gross Pay

```
hours = int(input("hours? "))
rate = float(input("rate? "))
if(hours > 80):
    gPay = 40 * rate
    gPay = gPay + 40 * (rate*1.5)
    gPay = gPay + (hours-80) * (rate*2.0)
elif(hours > 40):
    gPay = 40 * rate
    gPay = gPay + (hours-40) * (rate*1.5)
else:
    gPay = hours * rate
print("Gross Pay: ",gPay)
```

7

PORTLAND STATE UNIVERSITY

My First Try

```
hours = int(input("hours? "))
rate = float(input("rate? "))
if(hours > 80):
    gPay = 40 * rate
    gPay = gPay + 40 * (rate*1.5)
    gPay = gPay + (hours-80) * (rate*1.5)
elif(hours > 40):
    gPay = 40 * rate
    gPay = gPay + (hours-40) * (rate*1.5)
else:
    gPay = hours * rate
print("Gross Pay: ",gPay)
```

8

PORTLAND STATE UNIVERSITY

Code Inspection: *I thought I made a mistake once, but I was wrong*

- Always a good idea to look your code over for obvious mistakes
- If the main part of the code is too large to fit on a single screen, *print it out to paper*
- Peer inspections are commonly used
  - sit down and explain your program's execution to someone else – they probably won't find an error, but in the process of explaining it, you probably will

9

PORTLAND STATE UNIVERSITY

Testing to See If Your Program Works Correctly

- Testing is feeding your program some pre-determined inputs, **for which you have already calculated the expected results** to see if you get what you expected
- Possible Results:
  - **Correct** – actual results match your expected results
  - **Run-time Error** – program "crashes"
  - **Logic Error** – actual results do not match your expected results

10

PORTLAND STATE UNIVERSITY

Common Approach to Testing
case-based testing

- What are the various cases your program expects?
  - Worked 40 hours or less
  - Worked more than 40 hours but less than 80 hours
  - Worked more than 80 hours
- Your **test cases**:
  - hours = 30, rate = 12, expect: 360
  - hours = 45, rate = 12, expect: 570
  - hours = 90, rate = 12, expect: 1440

11

PORTLAND STATE UNIVERSITY

Specification Error

- What if the hours worked is 80 exactly?
  - *if hours worked is more than 80 … if hours worked is more than 40 but less than 80 …*
- Very common for the problem statement to be incorrect – if you were to implement this specification, your program would be technically correct, but the user would still not be happy
- Peer inspections work with specifications too

12

PORTLAND STATE UNIVERSITY

Logical Data Types

- Conditions evaluate to True or False:

```
if(hours > 80):
```

- You can have a variable that points to a location containing a True or False value

```
forever = True
while(forever):
    print("animal crackers")
```

13

PORTLAND STATE UNIVERSITY

Negation

- You can use **not** to "negate" a logical value:

```
forever = True
while(not forever):
    print("animal crackers")
```

- If x is True, **not x** is False
- if x is False, **not x** is True

14

PORTLAND STATE UNIVERSITY

Imply Logical Values from Numeric/String Values

- Zero is false
- The empty string “” is false
- Everything else is True

- ***Use sparingly – you should use explicit logical comparisons until you really understand what you are doing***

Compound Conditions

- Simple conditions usually consist of two values connected by a relational operator:

```
if(hours > 80):
```

- What if multiple conditions must be met?
- If the “*employee works over 80 hours **and** belongs to the widget transporter union they get double time for every hour worked over 80*”

Logical Operators

- `and`
- `or`
- logical operators connect two relational expressions

17

PORTLAND STATE UNIVERSITY

Check for hours over 80 *AND* a member of the union

```
unionQ = input("member of the union(Y/N)? ")
if(unionQ == "Y"):
    union=True
else:
    union=False
hours = int(input("hours? "))
rate = float(input("rate? "))
if((hours > 80)and(union)):
    gPay = 40 * rate
    gPay = gPay + 40 * (rate*1.5)
    gPay = gPay + (hours-80) * (rate*2.0)
else:
    gPay = hours * rate
print("Gross Pay: ",gPay)
```

18

PORTLAND STATE UNIVERSITY

Check for hours over 80 *AND* a member of the union - *alternate*

```
unionQ = input("member of the union(Y/N)? ")
hours = int(input("hours? "))
rate = float(input("rate? "))
if((hours > 80)and(unionQ == "Y")):
    gPay = 40 * rate
    gPay = gPay + 40 * (rate*1.5)
    gPay = gPay + (hours-80) * (rate*2.0)
else:
    gPay = hours * rate
print("Gross Pay: ",gPay)
```

19

PORTLAND STATE UNIVERSITY

Check for union membership, or boss' family member

```
union = input("member of union (Y/N)? ")
family = input("related to boss (Y/N)? ")
hours = int(input("hours? "))
rate = float(input("rate? "))
if((hours>80)and((union=="Y")or(family=="Y"))):
    gPay = 40 * rate
    gPay = gPay + 40 * (rate*1.5)
    gPay = gPay + (hours-80) * (rate*2.0)
else:
    gPay = hours * rate
print("Gross Pay: ",gPay)
```

20

PORTLAND STATE UNIVERSITY

Syntax of a logical expression

- {relational expr} {logical op} {relational expr}
- {logical op} is **and** or **or**
- Any place you can have a relational expression, you can have a logical expression
- if, while, etc.
- Logical expressions have an order to evaluation: left to right with **and** before **or**
- Use parentheses for grouping expressions – make sure they are balanced

21

PORTLAND STATE UNIVERSITY

Truth Tables

Relop1	Relop2	Relop1 and Relop2
T	T	T
T	F	F
F	T	F
F	F	F

Relop1	Relop2	Relop1 or Relop2
T	T	T
T	F	T
F	T	T
F	F	F

22

PORTLAND STATE UNIVERSITY

Truth Tables with *Negation*

Relop1	Relop2	not(Relop1 and Relop2)
T	T	F
T	F	T
F	T	T
F	F	T

Relop1	Relop2	not(Relop1 or Relop2)
T	T	F
T	F	F
F	T	F
F	F	T

PORTLAND STATE UNIVERSITY

Strings and Numerics

- input() always returns a string
- if we want to do arithmetic, we need to do a type conversion to numeric using int():

```
aNumber = int(input("enter number"))
```

- Works great if I enter “1234” but what if I type in “apple pie?”

PORTLAND STATE UNIVERSITY

Trying to Convert "apple pie" to a number – won't work

```
>>>
enter number apple pie
Traceback (most recent call last):
  File
  "C:/Users/Warren/Dropbox/Courses/CS161/Py
  thon Code/stringNints.py", line 1, in
  <module>
    aNumber=int(input("enter number "))
ValueError: invalid literal for int() with
  base 10: 'apple pie'
>>>
```

25

PORTLAND STATE UNIVERSITY

A Common Solution

- accept the input as a string
- check to see if the string is all numeric, if it is, convert it and if it isn't, display a message and ask the user to re-enter the value:

```
Get stringVariable
If stringVariable contains a number then
   convert string Variable to numeric and
   put in numericVariable
Else display an error message
```

26

PORTLAND STATE UNIVERSITY

isnumeric()

- a method() that can be applied to strings
- We use a *method* to apply an operation to an object such as a string.
- **`isnumeric()`** returns a True or False
- Syntax: **`stringVar.isnumeric()`**
- Semantics: return TRUE if stringVar holds a value that can be converted into a number, otherwise return FASLSE

27

PORTLAND STATE UNIVERSITY

Example

```
stringVar=input("enter number ")
if stringVar.isnumeric():
  print("you entered a number")
else:
  print("you did not enter a number")
  stringVar=input("enter number ")
print("the number is ",stringVar)
```

28

PORTLAND STATE UNIVERSITY

Revised Example Using Negation

```
stringVar=input("enter number ")
if not(stringVar.isnumeric()):
  print("you did not enter a number")
  stringVar=input("enter number ")
print("the number is ",stringVar)
```


PORTLAND STATE UNIVERSITY