Python
String Formatting
We really need to learn how to format strings

- Python has made several attempts before settling on an efficient syntax.
  - You can find information on the previous solutions on the net.
- Use the `format` function
  - Distinguish between the `blueprint`
  - and the `string to be formatted`
  - Result is the formatted string.
Formatting Strings

• Blueprint string
  • Uses {} to denote places for variables
• Simple example
  • "{} {}".format('one', 'two')

• Result ‘one two’
• Inside the brackets, we can put indices to select variables
  • 0 means first variable, 1 second, …
• Can reuse variables

```python
>>> "{0}, {0}, {1}, just {0}".format("great", "extraordinary")
'great, great, extraordinary, just great'
```
Formatting Strings

- Additional formatting inside the bracket after a colon
- Can assign the number of characters to print out

```python
>>> "\{0:10\}, \{1:10\}, \{0:10\}".format("funny", "nuts")
'funny     , nuts     , funny'
```

- Default alignment is to the left
Formatting Strings

- Use ^ to center
- Use < to left-align
- Use > to right-align

```python
>>> "{0:10}|{1:^10}|{0:>10}".format("sheep", "wolf")
'sheep    |    wolf    |    sheep'
```
Formatting Strings

• Numbers are handled without specifying format instructions.

```python
>>> "{} divided by {} is {} modulo {}".format(143, 29, 143//29, 143%29)
'143 divided by 29 is 4 modulo 27'
```

• Or we can insist on special types
  • Use s for string
  • Use d for decimal
  • Use f for floating point
  • Use e for floating point in exponential notation
Formatting Strings

- By specifying “f” we ask for floating point format
- By specifying “e” we ask for scientific format

```python
>>> "{0:f}, {0:e}".format(3.141)
'3.141000, 3.141000e+00'
```
• Padding

  • If the variable needs more space to print out, it will be provided automatically

>>> "{:10s}".format("Pneumonoultramicroscopicsilicovolcanoconiosis")
'Pneumonoultramicroscopicsilicovolcanoconiosis'

• This is actually the longest officially recognized word in English
• Padding:
  • On the reverse, we can give the number of significant digits after a period

```python
>>> "{:8.2f}".format(3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067982148086513282306647093844609550582231725359408128481)
' 3.14'
```

• We only want to keep two decimal digits after the period
• But use a total of 8 spaces for the number.
• Escaping curly brackets:

• If we want to write strings with format containing the curly brackets "{" and "}", we just have to write "{{{" and "}}}"

```python
>>> "{{{ {}, {} }}}}".format(3, 4)
'{ 3, 4 }'
```

• A single bracket is a placeholder, a double curly bracket is a single one in the resulting string.
Application: Pretty Printing

- Develop a mortgage payment plan
  - Accountants have formulae for that, but it is fun to do it directly
  - Assume you take out a loan of L$ dollars
    - The loan is financed at a rate of r% annually
    - Interest is paid monthly, i.e. at a rate of r/12%
  - Each month you make a repayment
    - Part of the repayment is to pay the interest
    - The remainder pays down the debt
Mortgage Payments

• Use a while-loop
  • Condition is that there is still an outstanding debt
  • Adjust outstanding debt
  • Count the number of payments
• Need to initialize values
Mortgage Payments

• We need values for:
  • Monthly Rate (interest in percent)/1200
  • Principal
  • Repayment
• Get those from the user
  • A true application would contain code that checks whether these numbers make sense.
Mortgage Payments

• Initialization

```python
princ = float(input("What is the principal? "))
rate = float(input("What is the interest rate (in percents)? "))/1200
print("Your minimum rate is ", rate*princ)
paym = float(input("What is the monthly payment? "))
month = 0
```
Mortgage Payments

- We continue until we paid down the principal to zero

```python
while princ > 0:
```
Mortgage Payments

- Update the situation in the while loop
- Last payment does not need to be full, so we calculate it

```python
intpaid = princ*rate
princ = princ + princ*rate - paym
if princ < 0:
    lastpayment = paym + princ
    princ = 0
month += 1
```
** The Ultimate Mortgage Calculator **

What is the principal 40000
What is the interest rate (in percents)? 4
Your minimum rate is 133.33
What is the monthly payment? 1950
This is what your mortgage scheme looks like

<table>
<thead>
<tr>
<th>Month</th>
<th>Interest</th>
<th>Principal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>133.33</td>
<td>38183.33</td>
</tr>
<tr>
<td>2</td>
<td>127.28</td>
<td>36360.61</td>
</tr>
<tr>
<td>3</td>
<td>121.20</td>
<td>34531.81</td>
</tr>
<tr>
<td>4</td>
<td>115.11</td>
<td>32696.92</td>
</tr>
<tr>
<td>5</td>
<td>108.99</td>
<td>30855.91</td>
</tr>
<tr>
<td>6</td>
<td>102.85</td>
<td>29008.76</td>
</tr>
<tr>
<td>7</td>
<td>96.70</td>
<td>27155.46</td>
</tr>
<tr>
<td>8</td>
<td>90.52</td>
<td>25295.98</td>
</tr>
<tr>
<td>9</td>
<td>84.32</td>
<td>23430.30</td>
</tr>
<tr>
<td>10</td>
<td>78.10</td>
<td>21558.40</td>
</tr>
<tr>
<td>11</td>
<td>71.86</td>
<td>19680.26</td>
</tr>
<tr>
<td>12</td>
<td>65.60</td>
<td>17795.86</td>
</tr>
<tr>
<td>13</td>
<td>59.32</td>
<td>15905.18</td>
</tr>
<tr>
<td>14</td>
<td>53.02</td>
<td>14008.20</td>
</tr>
<tr>
<td>15</td>
<td>46.69</td>
<td>12104.89</td>
</tr>
<tr>
<td>16</td>
<td>40.35</td>
<td>10195.24</td>
</tr>
<tr>
<td>17</td>
<td>33.98</td>
<td>8279.22</td>
</tr>
<tr>
<td>18</td>
<td>27.60</td>
<td>6356.82</td>
</tr>
<tr>
<td>19</td>
<td>21.19</td>
<td>4428.01</td>
</tr>
<tr>
<td>20</td>
<td>14.76</td>
<td>2492.77</td>
</tr>
<tr>
<td>21</td>
<td>8.31</td>
<td>551.08</td>
</tr>
<tr>
<td>22</td>
<td>1.84</td>
<td>0.00</td>
</tr>
</tbody>
</table>

You paid off the loan in 22 months, and your last payment was 552.92
Pretty-Printing Tables

• Format Strings revisited:
  • Format string — blueprint
  • Uses { } to denote spots where variables get inserted
Pretty-Printing Tables

• Syntax
  • \{a:^10.3f}\}
    • a — the number of the variable
    • Can be left out
    • : — what follows is the formatting instruction
    • 10 — number of spaces for the variable
    • . — what follows is the precision
    • 3 — precision
    • f — print in floating point format
Pretty-Printing Tables

- If the variable is larger than the space given:
  - Full value is printed out
  - Alignment by default is
    - left (<) for strings
    - right (>) for numbers
Pretty-Printing Tables

- Task:
  - A program that gives a table for the log and the exponential function between 1 and 10

- Hint: $x=1+i/10$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$\exp(x)$</th>
<th>$\log(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>2.71828</td>
<td>0.00000</td>
</tr>
<tr>
<td>1.10</td>
<td>3.00417</td>
<td>0.09531</td>
</tr>
<tr>
<td>1.20</td>
<td>3.32012</td>
<td>0.18232</td>
</tr>
<tr>
<td>1.30</td>
<td>3.66930</td>
<td>0.26236</td>
</tr>
<tr>
<td>1.40</td>
<td>4.05520</td>
<td>0.33647</td>
</tr>
<tr>
<td>1.50</td>
<td>4.48169</td>
<td>0.40547</td>
</tr>
<tr>
<td>1.60</td>
<td>4.95303</td>
<td>0.47000</td>
</tr>
<tr>
<td>1.70</td>
<td>5.47395</td>
<td>0.53063</td>
</tr>
</tbody>
</table>